Neural Wavelet Compression: Final Report

David Luo, Yu-An Chen
School of Computer Science, Carnegie Mellon University, USA
Department of Statistics and Data Science, Carnegie Mellon University, USA
{djluo, yuanche2}@andrew.cmu.edu

1 Introduction

Research into time-series compression methods is essential in today's data-driven world, where vast amounts of time-series data are generated continuously. Traditional storage solutions struggle to cope with the sheer volume of data being produced, leading to bottlenecks in data management systems. Moreover, slow data transfer speeds can hinder real-time analysis and decision-making processes, lowering businesses productivity and research output. Efficient compression not only reduces storage requirements, allowing organizations to store more data within existing resources, but it also facilitates faster data transfer, enabling quicker analysis and more timely insights. Existing time-series compression techniques are comprised of functional approximation algorithms, such as the Discrete Cosine/Wavelet Transform, as well as newer deep-learning based approaches, which generate hidden representations of time series with minimal assumptions. We seek to combine the mathematical underpinning of functional approximation algorithms with the flexibility of deep-learning architectures to create a Discrete Wavelet Transform based compression model that achieves state-of-the-art-results in decreasing reconstruction error for time series signals. We will validate our findings by comparing model performance on time-series data sets selected from the 4th Makridakis forecasting Competition. Our code is available at https://github.com/dluuo/NeuralWaveletCompression

2 Background

In the midway report, we discussed two main categories of compression algorithms: lossless and lossy. Lossless algorithms restore the original dataset with no error but are restricted in compression ratios. Lossy algorithms, which include functional approximation and autoencoder methods, trade some reconstruction error for lower compression ratios. We then detailed functional approximation techniques and autoencoder-based methods, noting the latter's lack of mathematical underpinning and interpretable models. With this in mind, we motivated implementing a model that builds upon existing autoencoder architectures by utilizing useful properties from functional approximation and wavelet literature to improve both the performance and interpretability of these models.

The baseline results were obtained using a simple time-series compression autoencoder. We evaluated the models based on compression ratios and reconstruction error (MSE and MAE). Two datasets were used: the Appliances Energy dataset and the Traffic Volume dataset. As shown in Figure 1, the results showed an inverse relationship between reconstruction error and compression ratio, indicating successful compression and decompression. With a baseline in place, we now seek methods that can demonstrate lower reconstruction error with similar compression ratios. However, we now note the datasets we used were not widely used as benchmarks of testing compression models, hence we transitioned to evaluating our methods on datasets from the 4th Makridakis Forecasting Competition.

3 Related Work

Our work builds upon the N-BEATS and N-HITS architectures, two state-of-the-art time series forecasting models.

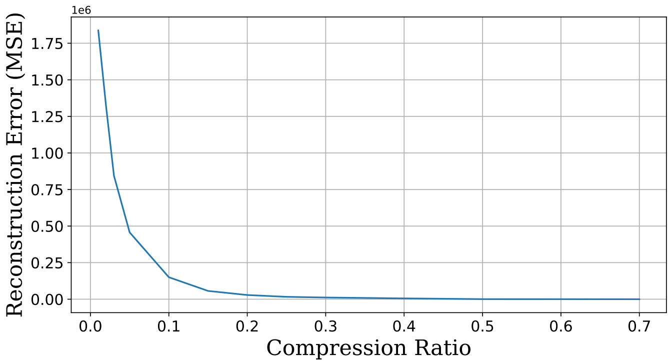

Figure 1: Test Reconstruction Error (MSE) vs Compression Ratio on the TV Dataset for the Linear Baseline Autoencoder.

The N-BEATS (Neural Basis Expansion Analysis for Time Series) architecture is able to decompose the input time series by performing separate local nonlinear projections onto different basis functions, in particular sines and cosines, across different blocks. Each block has a fully connected layer, and the blocks are grouped in "stacks", where each stack corresponding to specific basis functions. This allows each part of the model to focus on specific frequencies from the original input signal, both improving predictive accuracy and interpretability [Oreshkin et al., 2020]. As a result, we modeled our Discrete Cosine Transform autoencoder off the N-BEATS architecture.

The N-HITS (Neural Hierarchical Interpolation for Time Series) architecture is similar to the N-BEATS architecture in that it performs nonlinear projections onto basis functions. However, it shifts from using sine/cosine functions to using wavelet theory, allowing it to handle a wider range of time series, especially those with abrupt changes or non-periodic components. Wavelets, known for being both time and frequency sensitive, enable NHITS to decompose time series data into multi-resolution components. [Challu et al., 2023] We modeled our Discrete Wavelet Transform off the N-HITS architecture.

4 Methodology: Neural Wavelet Compression

4.1 Theoretical Background

We build our *Neural Wavelet Compression* algorithm on the advantageous theoretical guarantees of the wavelet transform [Daubechies, 1992].

Neural Basis Approximation Theorem. Let a time series signal $\mathcal{Y}(\cdot) : [0,1]^L \to \mathbb{R}$, be a square-integrable function $\mathcal{Y} \in \mathcal{L}^2([0,1])$. Let $V_w = \{\phi_{w,h}(\tau) = \phi(2^w(\tau - h)) \mid w \in \mathbb{Z}, h \in 2^{-w} \times [0,\ldots,2^w]\}$, denote a multi-resolution function space with arbitrary approximation capabilities to $\mathcal{L}^2([0,1])$. If the projection $\mathrm{Proj}_{V_w}(\mathcal{Y}(\tau))$ varies smoothly on its observed points $\mathbf{y}_{[:t]}$, then the time series signal $\mathcal{Y}$ can be arbitrarily approximated by a neural network that learns a finite number of multi-resolution coefficients $\hat{\theta}_{w,h}$. That is $\forall \epsilon > 0$,

$$\int |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| d\tau = \int |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\theta}_{w,h}\phi_{w,h}(\tau)| d\tau \leq \epsilon \tag{1}$$

Examples of multi-resolution functions $V_w = \{\phi_{w,h}(\tau)$ include piece-wise constants and piece-wise linear functions. A proof of the Neural Basis Approximation Theorem is available in Appendix A.

4.2 Model Architecture

The Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT) Autoencoders had similar architectures, both comprised of two main components: encoder and decoder. A diagram of the architecture can be seen in Figure 3.

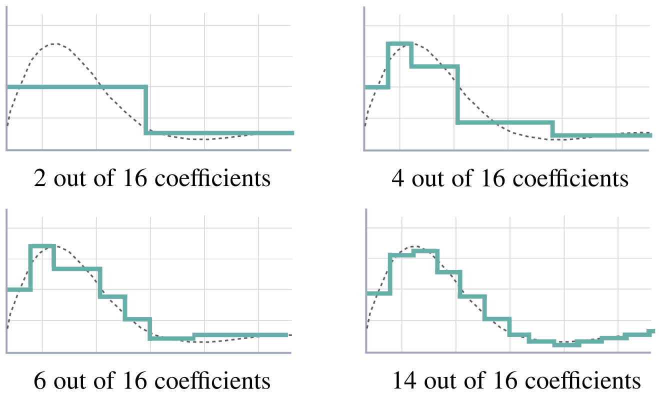

Figure 2: An example of wavelet compression on a time series. Our algorithm can arbitrarily approximate the signal by learning the multi-resolution coefficients $\hat{\theta}_{w,h}$ using a neural network.

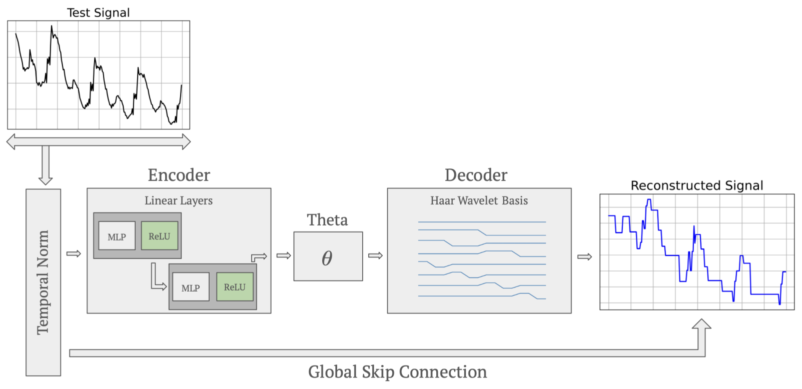

Figure 3: Discrete Wavelet Transform Model Architecture. The encoder outputs a wavelet coefficient matrix, which is combined with the Haar Wavelet Basis in the decoder to reconstruct the signal.

The encoder transforms the input data into a lower-dimensional matrices that can be used by the decoder. The encoder returns returns a theta (coefficient) matrix, which will be used along with the cosines/wavelet basis in the decoder to reconstruct the test signal. The DCT Encoder also returns an additional shift matrix, which allows the model to also learn the proper shift for the cosine basis. For both models, the encoder is comprised of multiple layers, with each layer consisting of a linear module followed by an activation function. For our experiments, we instantiated the encoders with two layers each with 128 hidden size, and the ReLU activation function. If the input time series is multivariate, the encoder will compress each dimension independently.

The DWT and DCT decoders construct wavelet or cosines basis matrices respectively, and then multiply them by the compressed coefficient matrices in order to reconstruct the original signal. For the DCT decoder, the model generates a matrix comprised of series of cosine functions with varying frequencies, that is shifted with the shift matrix returned my the DCT encoder. For our experiments, we chose to generate a cosines basis matrix comprised of frequencies from 0 to 100. The the DWT decoder, the model generates a wavelet basis matrix, where each row of the matrix corresponds to a distinct haar wavelet. For our experiments, we generated a Haar wavelet basis matrix with 256 distinct wavelets, each length 1000.

The autoencoders are also able to be outfitted with a scaler/temporal normalization layer before the data is passed into the autoencoder. The scaler layer normalizes the input data through the following transformation:

$$\hat{\mathbf{x}}_{[i][t][c]} = \frac{(\mathbf{x}_{[i][t][c]} - \bar{\mathbf{X}}_{[i][c]})}{\hat{\sigma}_{[i][c]}} \tag{2}$$

Where $\bar{\mathbf{X}}_{[i][c]})$, $\hat{\sigma}_{[i][c]}$ are the mean and std of the training set, respectively. The inverse transformation will then be applied to the output of the decoder. For the test set, the inverse transformation will be performed with the mean and std of the training set.

5 Results

In this section, we compare the performance of the Discrete Wavelet Transform (DWT) Autoencoder against the Disrete Cosine Transform (DCT) and Linear Autoencoders. We will first detail the datasets and evaluation metrics used in the experiment. as well as two ablation studies we conducted on the effects on L1 Norm and Trend. Then, we show the results of the main experiments, and detail our learnings from the results.

5.1 Datasets

For our midway report, we chose two datasets used in a previous time series compression papers: Appliances Energy (AE) dataset [Candanedo, 2017], and Traffic Volume (TV) dataset [Hogue, 2019].

In this report, we evaluated our models on the 4th Makridakis Forecasting Competition (M4) dataset [Makridakis et al., 2020], which is a much more popular dataset consisting of many individual time series, split into monthly, weekly, daily, and other frequencies. For our experiments, we selected the monthly, weekly, and daily frequencies, and (due to computational constraints) only kept series between 2500 and 3000 in length. As a result, our train/test data consisted of 94 time series at the daily level, 2 time series at the weekly level, and 4 time series at the monthly level. Training our models on multiple time series allowed use to evaluate their performance on a wider variety of signals, and gives us more robust performance metrics.

For each series, we set aside the last 1000 time steps for the test set, and sampled rolling windows of size 1000 with stride 20 on remaining time steps to construct the train set.

5.2 Evaluation Metrics

We use the compression ratio alongside the reconstruction error to evaluate our autoencoders. The compression ratio measures the method's capacity to reduce the memory footprint of a dataset, indicating how efficiently it can reduce the data size. On the other hand, reconstruction error measures a method's proficiency in reconstructing the original dataset from the compressed version, highlighting the accuracy of the compression-decompression process.

Consider an original time series signal $\mathbf{y}_{[:t]} \in \mathbb{R}^T$, and its compressed version $\hat{\mathbf{y}}_{[:t]} \in \mathbb{R}^T$. The compression ratio ρ of a technique is defined as:

$$\rho = \frac{\text{size}(\hat{\mathbf{y}}_{[:t]})}{\text{size}(\mathbf{y}_{[:t]})} \tag{3}$$

To enforce the compression ratio during test time, our autoencoders remove the coefficients with the smallest magnitude before passing them into the decoder. To the measure the reconstruction proficiency of our methods on the test set, we use the *mean absolute error* (MAE) between the true signal and the reconstructed signal:

$$\text{MAE} = \frac{1}{NTD}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{d=1}^{D}|y_{\text{pred},i,t,d} - y_{\text{test},i,t,d}| \tag{4}$$

During train time, our autoencoders used the HuberLoss:

$$\text{HuberLoss} = \frac{1}{NTD}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{d=1}^{D}\begin{cases}\frac{1}{2}(y_{\text{pred},i,t,d} - y_{\text{test},i,t,d})^2 & \text{for } |y_{\text{pred},i,t,d} - y_{\text{test},i,t,d}| \leq \delta, \\ \delta\left(|y_{\text{pred},i,t,d} - y_{\text{test},i,t,d}| - \frac{1}{2}\delta\right) & \text{otherwise.}\end{cases} \tag{5}$$

along with L1 Norm. We chose the HuberLoss due to its robustness and smoothness properties. The Huberloss behaves like the L1 loss for larger values, so it penalizes alrger losses less harshly, and behaves like the L2 loss around 0, whose smoothness leads to more reliable gradient descent updates. We used the default PyTorch HuberLoss configuration, which has (δ = 1).

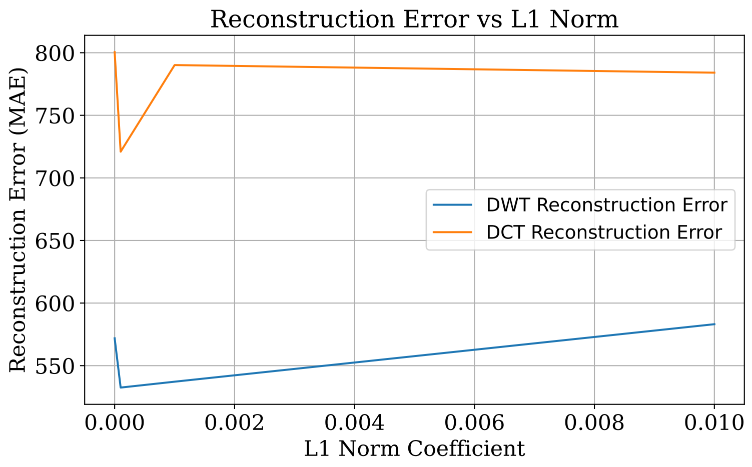

Figure 4: Test reconstruction error vs L1 Norm Coefficient λ on Monthly M4 Dataset. Initially, as λ increases, the model test performance improves and reaches an optimal value at $\lambda = 0.0001$, but subsequently deteriotates as λ gets larger.

5.3 L1 Norm Ablation Study

During training, the DWT and DCT autoencoders are trained with the following loss:

$$\text{Loss} = \text{HuberLoss}(y_{\text{true}}, y_{\text{pred}}) + \lambda \frac{1}{NM} \sum_{i=1}^{N} \sum_{k=1}^{M} |y_{\text{pred},i,k}| \tag{6}$$

Where λ is the L1 Norm coefficient. The L1 Norm coefficient allows the model to regularize/zero out many of the coefficients in the compressed theta matrix, so the model can focus on retrieving only the most important coefficients. To find the optimal L1 Norm coefficient λ, we conducted an ablation study where we trained our model with $\lambda \in (0, 0.0001, 0.001, 0.01)$ on the Monthly M4 Dataset, and evaluated our model's reconstruction error on test data. The results, as seen in Figure 4, show a U-shaped curve, which is indicative of a bias-variance tradeoff with respect to λ. Initially, as λ increases, the model test performance improves and reaches an optimal value at $\lambda = 0.0001$. From there, the test performance deterioates as λ gets larger. This shows that encouraging the encoder to output sparse coefficient matrices helps improve model reconstruction performance, but it should not be the main focus of model. As a result of this study, we chose $\lambda = 0.0001$ in our main table experiments.

5.4 Stationarity Ablation Study

To see the effects of non-stationarity/trend in the input data on the reconstruction error of our models, as well as the effect of the Scaler/TemporalNormalization module, we tested the DWT and DCT autoencoders on an artifical time series signal of the following form:

$$\text{signal}(t) = 5\sin(3t) + 10\sin(6t) + \epsilon(t) + \beta \cdot t \tag{7}$$

Where $\epsilon(t) \sim N(0, 0.5)$ is random noise, and β is the trend coefficient. When $\beta = 0$, the time series is stationary, and as β increases, the time series becomes increasingly non-stationary. We trained four autoencoders: DWT without scalar), DWT (with scalar), DCT (without scalar), and DCT (with scalar) on the artificial time series signal for $\beta \in (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7)$.

From Figure 5, we see that for stationary time series (no trend), the DCT Autoencoder performs extremely well, and outperforms the DWT Autoencoder. However, as the trend coefficient increases, the DCT Autoencoder performance deteriorates quickly, while the DWT Autoencoder performance stays the same and outperforms it.

This shows how the DWT Autoencoder greatly outperforms the DCT Autoencoder in the presence of non-stationary time series. This makes sense, since the DWT decomposes a signal into a set of

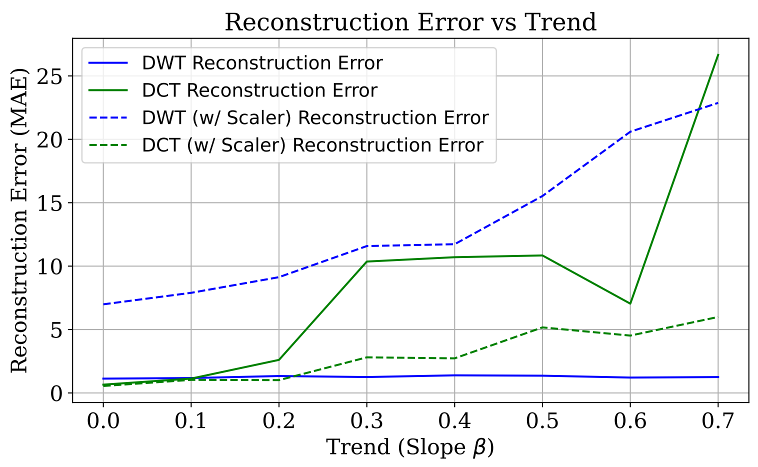

Figure 5: An example of wavelet compression on a time series. Our algorithm can arbitrarily approximate the signal by learning the multi-resolution coefficients $\hat{\theta}_{w,h}$ using a neural network.

wavelets, which are localized in both time and frequency. This allows DWT Autoencoder to adapt to changes in the signal, making it more efficient in representing non-stationary components like trend, or sudden shifts. In contrast, the DCT is more suited for signals with a stationary/periodic nature. DCT works by transforming the signal into a set of cosine functions, each of which are global and not localized in time. As a result, it is less capable of adapting to non-stationary signals, whose characteristics change over time.

However, when we add a scaler/temporal normalization layer to the DWT autoencoder (as described in section 4.2), the model performs much better for the varying trend levels. In Figure 5, we see that the DWT Autoencoder with scaler starts off with good performance, and only deteriorates in performance slightly for the higher trend levels. This is because the scaler layer reduces the impact of large-scale changes or shifts in the data, allowing the model to focus more on the underlying patterns rather than the scale of the data. Interestingly, the DCT Autoencoder performs much worse when augmented with a temporal normalization layer.

As a result of this ablation study, we chose to use a Scaler layer for the DCT Autoencoder for the main table experiments, but did not include a Scaler layer for the DWT Autoencoder.

5.5 Main Table Hyperparameters

For the main table experiments, we used trained all three autoencoders with 0.001 learning rate over 500 epochs. As a result of the L1 Norm and Trend Ablation studies, we trained both the DWT and DCT Autoencoders with 0.0001 L1 Norm coefficients, and added a scaler/temporal normalization layer to the DWT Autoencoder.

For all three autoencoders, we tested their reconstruction error for compression ratios 0.01, 0.02, 0.03, 0.04, 0.05.

5.6 Example Reconstructions

In Figure 6, we show an example test signal from the M4 dataset (daily frequency), and the corresponding reconstructed signals from the DWT and DCT autoencoders. Despite the test signal from the M4 dataset being quite non-stationary, both the DWT and DCT Autoencoders are able to reliably reconstruct the test signal.

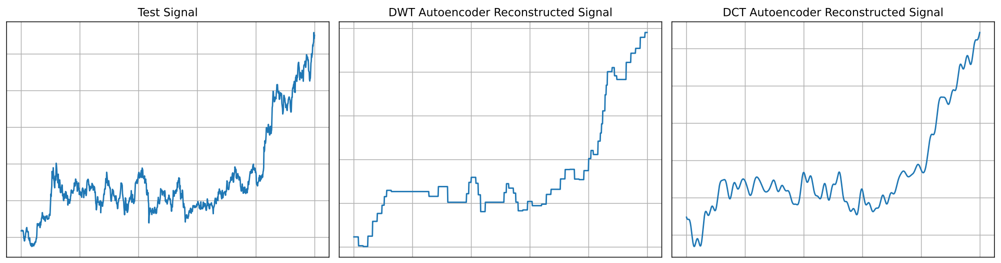

Figure 6: A visual comparison of the reconstructed signals from the DWT and DCT Autoencoders against the Test Signal.

5.7 Main Table Results

Dataset	Compression Ratio	DWT Autoencoder	DCT Autoencoder	Linear Autoencoder
M4 - Monthly	0.01	**564.603**	574.953	755.584
M4 - Monthly	0.02	**536.764**	593.868	741.286
M4 - Monthly	0.03	**540.430**	612.616	559.032
M4 - Monthly	0.04	**540.569**	613.604	575.450
M4 - Monthly	0.05	**542.000**	621.818	558.912
M4 - Weekly	0.01	**562.914**	571.007	711.433
M4 - Weekly	0.02	540.429	**461.356**	622.841
M4 - Weekly	0.03	508.716	**426.451**	485.082
M4 - Weekly	0.04	483.279	462.759	**441.323**
M4 - Weekly	0.05	437.294	518.230	**403.105**
M4 - Daily	0.01	164.876	170.279	**153.581**
M4 - Daily	0.02	**126.109**	128.092	126.819
M4 - Daily	0.03	**111.904**	112.946	127.391
M4 - Daily	0.04	**106.448**	107.866	114.006
M4 - Daily	0.05	103.518	105.369	**91.796**

6 Discussion and Analysis

From the main table results, we observe the the Discrete Wavelet Transform Autoencoder significantly outperforms all other methods at all the given compression ratios for the monthly time series, and performs moderately well for the weekly and daily time series. This is expected, as we have shown previously that wavelets have useful theoretical guarantees that along them to approximate many time series functions, including signals that are non-stationary.

We also see that at lower compression ratios, both the DWT and DCT Autoencoders perform relatively well when compared against the Linear Autoencoder. However, as the compression ratio increases, DWT and DCT Autoencoders reconstruction error are relatively flat, while the Linear Autoencoder's performance steadily improves. We believe this happens due to the robustness of the DWT/DCT Autoencoders, as well as the flexibility of the Linear Autoencoder. At lower compression ratios, when the compressed theta is small, both the DWT/DCT Autoencoders are able to focus on only the most important coefficients and discard the unimportant coefficients. Since the majority of the signal can be encoded in the largest magnitude coefficients, the DWT and DCT perform well compared to the Linear Autoencoder. Linear Autoencoders, being simpler in their approach, might initially struggle to efficiently capture the complexity of the data at lower compression ratios

However, as the compression ratio increases, the compressed data size is large enough for the Linear Autoencoder to reliably reconstruct the signal. In addition, since the Linear Autoencoder is not restricted to using a cosine/wavelet basis in the reconstruction step, it is more flexible than the DWT/DCT autoencoders, and can potentially reconstruct more minute details in the signal. On the other hand, the DWT/DCT autoencoders may not be able to reconstruct every aspect of the signal when restricted to a cosine/wavelet basis, leading to their plateau in reconstruction error.

6.1 Future Work

Here, we consider initiatives that can extend upon and or consolidate our findings. Specifically:

1. For the Wavelet-Based Autoencoder, we used the Haar wavelet given its simplistic nature. Further research can attempt using other wavelets as the basis of time-series decomposition. For instance, other wavelets such as Daubechies or Meyer Wavelets can be used to conduct the necessary decompositions. As an even further extension, one can try learning the wavelet's structure instead of predetermining the specific wavelets.
2. Explore more complicated architectures in our autoencoders, since our wavelet models currently only use linear layers to learn the relevant coefficients. Further research can attempt to apply LSTM and or Transformer architectures to learn the coefficients, which can not just potentially improve coefficient learning performance but also enable the models to encode arbitrary length sequences. This can definitely be useful since this feature would eliminate the need to train new autoencoders each time our input time series length changes.
3. Explore different metrics for evaluating model performance. We currently evaluate test performance MAE, and train our model with a combination of Huber Loss and L1 Norm regularization. However, other metrics such as Root Mean Squared Error, Mean Absolute Percentage Error, and Mean Squared Logarithmic Error can be used to evaluate model performance depending on the use case of the models.

References

Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

Albert Boggess and Francis J Narcowich. *A first course in wavelets with Fourier analysis*. John Wiley & Sons, 2015.

Luis Candanedo. Appliances energy prediction. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5VC8G.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. NHITS: Neural Hierarchical Interpolation for Time Series forecasting. In *The Association for the Advancement of Artificial Intelligence Conference 2023 (AAAI 2023)*, 2023. URL `https://arxiv.org/abs/2201.12886`.

Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width, 2017. URL `https://arxiv.org/abs/1710.11278`.

John Hogue. Metro Interstate Traffic Volume. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5X60B.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4 (2):251–257, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL `https://www.sciencedirect.com/science/article/pii/089360809190009T`.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL `https://www.sciencedirect.com/science/article/pii/S0169207019301128`. M4 Competition.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting, 2020.

A Appendix

A.1 Neural Basis Approximation Theorem

This appendix will prove the approximation theorem introduced in Section 4. We will prove it using a two-stage approximation strategy, first showing that the wavelet basis can approximate any reasonably smooth signal and then proving that if the wavelet coefficients are smooth concerning the signal, a neural network inherits the approximation capabilities, too. We adapt this theorem from [Challu et al., 2023] 's work for signal reconstruction.

Lemma 1. Let the time series signal be $\mathcal{Y} : [0, 1] \to \mathbb{R}$ a square integrable function $\mathcal{L}^2([0, 1])$. The signal $\mathcal{Y}$ can be arbitrarily well approximated by a linear combination of piecewise constants:

$$V_w = \{\phi_{w,h}(\tau) = \phi(2^w(\tau - h)) \mid w \in \mathbb{Z}, h \in 2^{-w} \times [0, \ldots, 2^w]\}$$

where $w \in \mathbb{N}$ controls the frequency/indicator's length and h the time-location (knots) around which the indicator $\phi_{w,h}(\tau) = \mathbb{1}\{\tau \in [2^{-w}(h-1), 2^{-w}h]\}$ is active. That is, $\forall \epsilon > 0$, there is a $w \in \mathbb{N}$ and $\hat{\mathcal{Y}}(\tau|\mathbf{y}_{[:t]}) = \mathrm{Proj}_{V_w}(\mathcal{Y}(\tau|\mathbf{y}_{[:t]})) \in \mathrm{Span}(\phi_{w,h})$ such that

$$\int_{[0,1]} |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| d\tau = \int_{[0,1]} |\mathcal{Y}(\tau) - \sum_{w,h} \theta_{w,h}\phi_{w,h}(\tau)| d\tau \leq \epsilon \tag{8}$$

Proof. This classical proof can be traced back to Haar's work (1910). The indicator functions $V_w = \{\phi_{w,h}(\tau)\}$ are also referred in literature as Haar scaling functions or father wavelets. Details are provided in Boggess and Narcowich [2015].Let the number of coefficients for the ϵ-approximation $\hat{\mathcal{Y}}(\tau|\mathbf{y}_{[:t]})$ be denoted as $N_\epsilon = \sum_{i=0}^{w} 2^i$. □

Lemma 2. Let a time series signal $\mathcal{Y}(\cdot \mid \mathbf{y}_{[:t]}) : [0, 1]^L \to \mathcal{L}^2([0, 1])$ be ϵ-approximated by $\hat{\mathcal{Y}}(\tau|\mathbf{y}_{[:t]}) = \mathrm{Proj}_{V_w}(\mathcal{Y}(\tau|\mathbf{y}_{[:t]}))$, the projection to multi-resolution piecewise constants. If the relationship between the time series observations $\mathbf{y}_{[:t]} \in [0, 1]^T$ and wavelet coefficients $\theta_{w,h}$ varies smoothly, for instance $\theta_{w,h} : [0, 1]^L \to \mathbb{R}$ is a K-Lipschitz function then for all $\epsilon > 0$ there exists a three-layer neural network $\hat{\theta}_{w,h} : [0, 1]^T \to \mathbb{R}$ with $O\left(L(\frac{K}{\varepsilon})^L\right)$ neurons and ReLU activations such that

$$\int_{[0,1]^L} |\theta_{w,h}(\mathbf{y}_{[:t]}) - \hat{\theta}_{w,h}(\mathbf{y}_{[:t]})| d\mathbf{y}_{[:t]} \leq \epsilon \tag{9}$$

Proof. This lemma is a particular case of the neural universal approximation theorem that states the approximation capacity of neural networks of arbitrary width [Hornik, 1991]. The theorem has refined versions where the width can be decreased under more restrictive conditions for the approximated function [Barron, 1993, Hanin and Sellke, 2017]. □

Neural Basis Approximation Theorem. Let a time series signal $\mathcal{Y}(\cdot) : [0, 1]^L \to \mathbb{R}$, be a square-integrable function $\mathcal{Y} \in \mathcal{L}^2([0, 1])$. Let $V_w = \{\phi_{w,h}(\tau) = \phi(2^w(\tau - h)) \mid w \in \mathbb{Z}, h \in 2^{-w} \times [0, \ldots, 2^w]\}$, denote a multi-resolution function space with arbitrary approximation capabilities to $\mathcal{L}^2([0, 1])$. If the projection $\mathrm{Proj}_{V_w}(\mathcal{Y}(\tau))$ varies smoothly on its observed points $\mathbf{y}_{[:t]}$, then the time series signal $\mathcal{Y}$ can be arbitrarily approximated by a neural network that learns a finite number of multi-resolution coefficients $\hat{\theta}_{w,h}$. That is $\forall \epsilon > 0$,

$$\int |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| d\tau = \int |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\theta}_{w,h}\phi_{w,h}(\tau)| d\tau \leq \epsilon \tag{10}$$

Proof. For simplicity of the proof, we will omit the conditional lags $\mathbf{y}_{[:t]}$. Using both the neural approximation $\tilde{\mathcal{Y}}$ from Lemma 2, and Haar's approximation $\hat{\mathcal{Y}}$ from Lemma 1,

$$\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau = \int |(\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)) + (\hat{\mathcal{Y}}(\tau) - \tilde{\mathcal{Y}}(\tau))| d\tau$$

By the triangular inequality:

$$\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq \int |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| + |\sum_{w,h} \theta_{w,h}\phi_{w,h}(\tau) - \sum_{w,h} \hat{\theta}_{w,h}\phi_{w,h}(\tau)| d\tau$$

By a special case of Fubini's theorem

$$\begin{aligned}&\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq \\ &\int |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} \int_{\tau} |(\theta_{w,h} - \hat{\theta}_{w,h})\phi_{w,h}(\tau)| d\tau\end{aligned}$$

Using positivity and bounds of the indicator functions

$$\begin{aligned}&\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq \\ &\int_{\tau} |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} |\theta_{w,h} - \hat{\theta}_{w,h}| \int_{\tau} \phi_{w,h}(\tau) d\tau \\ &\qquad < \int_{\tau} |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} |\theta_{w,h} - \hat{\theta}_{w,h}|\end{aligned}$$

To conclude we use both arbitrary approximations from the Haar projection and the approximation to the finite multi-resolution coefficients

$$\begin{aligned}&\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq \\ &\int |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} |\theta_{w,h} - \hat{\theta}_{w,h}| \leq \epsilon_1 + N_{\epsilon_1}\epsilon_2 \ \leq \epsilon\end{aligned}$$

□