Structural Complexity in Synthetic Time Series

April 21, 2025

1 Structural Complexity in Synthetic Time Series: A Lightweight Framework

1.1 Background

Multivariate forecasting has wide applications across domains — including demand prediction, user activity tracking, predictive maintenance, risk modeling, stock movement forecasting, and patient monitoring, and many other fields and tasks.

But choosing the right forecasting model depends heavily on two things:

- The **underlying data-generating process**
- The **inductive biases** of the model (i.e., what structural patterns it learns easily)

For example:

- **Temporal Convolutional Networks (TCNs)** are biased toward short-term, local dependencies.
- **Vector AutoRegression (VAR)** assumes linearity and fixed lag structures.
- **Transformers** can learn long-range interactions, but may struggle with high-frequency chaos unless tuned properly.

Even if you suspect certain causal patterns in your data, there's often no way to know for sure—outside of your model's own conclusions.

1.2 The Central Question:

What makes one time series harder to model than another — in a way that impacts model choice?

Our goal is to provide a structured and reproducible framework for generating synthetic time series with known causal dependencies, and for quantifying the resulting signal complexity using:

- **Lag topology metrics** (depth, entropy, irregularity)
- **Spectral complexity** (via FFT and entropy)
- **Composite structural complexity score**

The goal is to better understand the relationship between time series structure and the inductive biases or capabilities required of neural architectures (e.g., Transformers, TCNs, RNNs). Inductive biases refer to the built-in assumptions of a model — such as locality, sequentiality, or global attention — that influence what kinds of temporal patterns it learns easily.

1.3 1. Synthetic Time Series Generator

We define a modular function to generate synthetic data with controllable lag structures: - Specify `causal_lags = [l1, l2, ..., ln]` - Assign weights `w = [w1, ..., wn]` to lagged values - Generate target variable (e.g., `close`) using a combination of lagged endogenous and exogenous features + noise

Key parameters: - `noise_std`: controls signal-to-noise ratio - `warm_start_mean`: stabilizes early values - `volume_scale`, `trade_count_scale`: modulate exogenous impact - `clip_price`: avoids numerical instability

This gives full control over the **temporal memory structure** and lets us systematically vary configurations.

1.4 2. Lag Topology Metrics

After the data is generated (again, with known temporal memory structure created by the user), we can measure and combine a few metrics for a composite complexity score.

1.4.1 a. Weighted Lag Depth

Measures the effective memory horizon (essentially a combination of how far back and how impactful a lag impact is):

$$\text{Depth} = \sum w_i \cdot \ell_i$$

Normalized across all configs:

$$\text{Depth}_{\text{norm}} = \frac{\sum w_i \cdot \ell_i}{\max(L)}$$

1.4.2 b. Lag Entropy

Measures how focused or diffuse the lag weights are:

$$H = -\sum w_i \log_2 w_i$$

Normalized by:

$$\log_2(n)$$

1.4.3 c. Lag Gap Entropy

Measures irregularity in temporal spacing:

$$g_i = \ell_{i+1} - \ell_i \quad \text{(for sorted lags)}$$

$$p_i = \frac{g_i}{\sum g_j} \quad H_{\text{gap}} = -\sum p_i \log_2 p_i$$

Assign:

$$H_{\text{gap}} = 0$$

if only one lag is present.

1.5 3. Spectral Entropy

Use FFT to compute the power spectrum (this is a standard metric option for understanding complexity in a time series and we are using it here as an additional metric to, in a way, control for that inherent complexity outside of the lag-causality metrics above):

1. Demean the signal: $x' = x - \bar{x}$
2. Take FFT: $\text{fft} = \text{FFT}(x')$
3. Compute power: $P = |\text{fft}|^2$
4. Normalize: $p_i = \frac{P_i}{\sum P}$
5. Compute entropy: $H_s = -\sum p_i \log_2 p_i$

High $H_s \to$ broad energy distribution (chaotic signal)
Low $H_s \to$ few dominant frequencies (structured/periodic)

1.6 4. Composite Structural Complexity Score

Combines signal and memory structure into one measure:

$$C = \alpha D_{\text{norm}} + \beta H_{\text{lag}} + \gamma H_{\text{gap}} + \delta H_{\text{spec}}$$

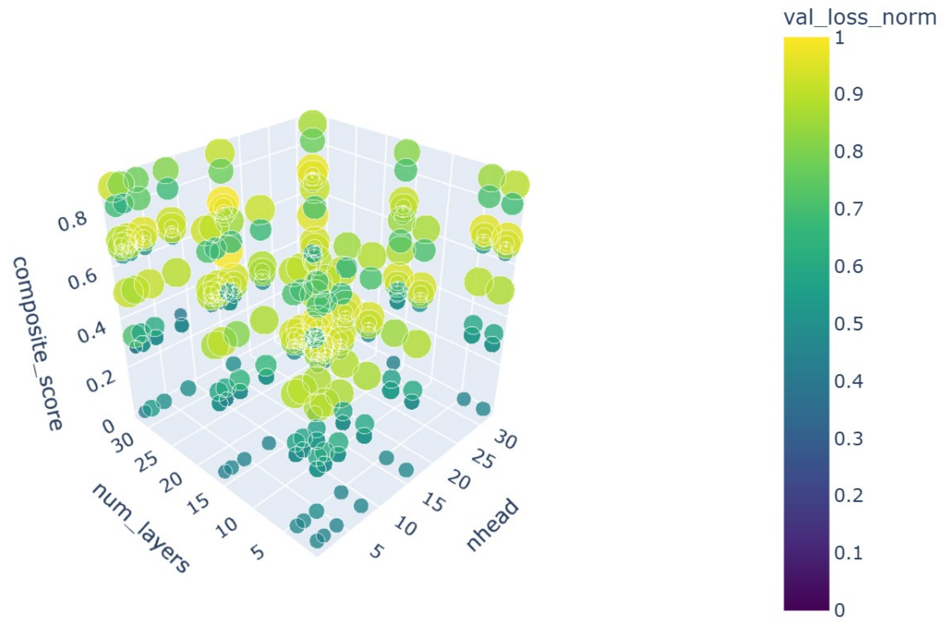
Where:
- D_{norm}: normalized weighted lag depth
- H_{lag}: lag entropy (normalized)
- H_{gap}: gap entropy (normalized)
- H_{spec}: spectral entropy (from FFT power spectrum)

The weights α, β, γ, and δ allow you to prioritize different structural traits.

A higher composite score suggests: - Longer effective memory (D_{norm}) - Greater diversity in memory sources (H_{lag}) - More irregular timing of influences (H_{gap}) - More spectral spread or chaos in the signal (H_{spec})

1.7 Example Output

Here is a 3-D scatterplot looking at composite complexity score vs. a few Transformer architectures (with causal masking, attention, etc.). You can see that as the composite complexity score increases, validation loss increases (while holding signal-to-noise ratio constant). Although it is difficult to see in a static plot here, increasing the number of attention heads and layers of the network does improve validation loss (within each 'level' of composite complexity score).


val_loss_norm
composite_score
num_layers
nhead