Emergent Latent-State Computation under Stochastic Volatility

Lulu Wang[1], Xiaoyu Huang[2]

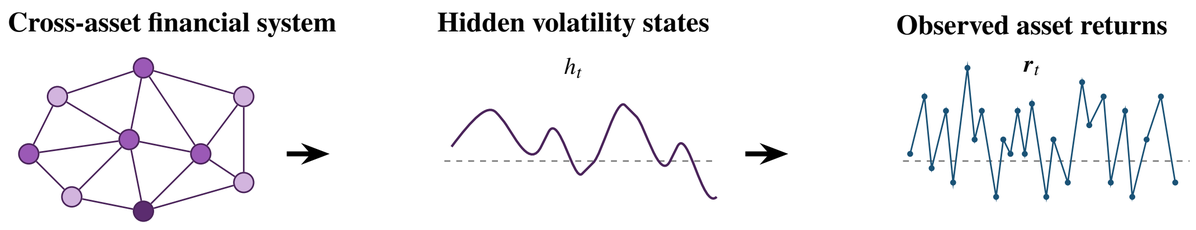
[1]*Department of Data Analytics, Dickinson College* [2]*Department of Mathematics, Temple University*

1. Motivation

Mechanistic interpretability has primarily focused on language models and deterministic synthetic tasks. We study whether it can also support **financial knowledge discovery** by revealing economically meaningful latent structure learned from noisy market observations. **Stochastic volatility** provides a controlled benchmark for testing whether neural sequence models internally recover hidden volatility states and can support knowledge discovery.

Controlled benchmark

- Stochastic-volatility models treat observed returns as **noisy measurements** of an underlying latent volatility state. This creates a setting that is both economically meaningful and experimentally controlled: the model is trained only on returns, while the true latent state is available to the researcher in simulation.



Stochastic volatility models treat returns as noisy observations of an underlying latent stochastic state.

Mechanistic hypotheses

1. **Is the latent state encoded?**
   The model's hidden representation may contain information about the next latent volatility state $\boldsymbol{h}_{t+1}$.
2. **Where does it emerge?**
   Latent-state information may become available at specific architectural stages, such as input normalization, positional embeddings, attention, or feed-forward layers.
3. **What computation extracts it?**
   The model may implement an implicit filtering computation that transforms past returns into an internal estimate of the latent volatility state.

2. Problem Formulation

Multivariate stochastic volatility process

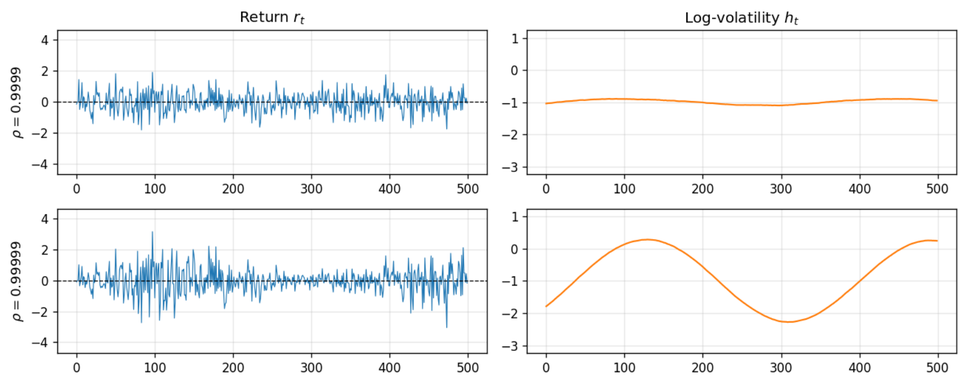
- Observed returns:

$$\boldsymbol{r}_t = \exp(\boldsymbol{h}_t/2) \odot \boldsymbol{\varepsilon}_t, \qquad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \Sigma_\varepsilon).$$

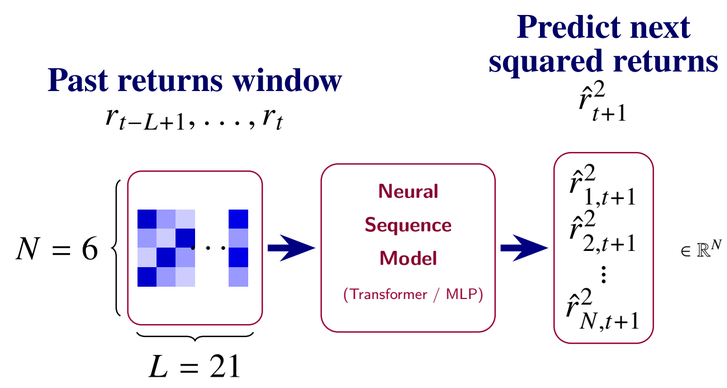
- Latent state:

$$\boldsymbol{h}_{t+1} = \boldsymbol{\mu} + A_\omega(\boldsymbol{h}_t - \boldsymbol{\mu}) + \boldsymbol{\eta}_t, \qquad \boldsymbol{\eta}_t \sim \mathcal{N}(0, \Sigma_\eta).$$

  - $\boldsymbol{h}_t$: latent log-volatility state for $N = 6$ asset
  - $\boldsymbol{r}_t$: observed noisy returns
  - A_ω: induces cross-asset spillovers



Supervised learning task



The model observes $L = 21$ past returns for $N = 6$ assets and predicts

$$y = (r^2_{1,t+1}, \ldots, r^2_{N,t+1})^\top.$$

Training losses

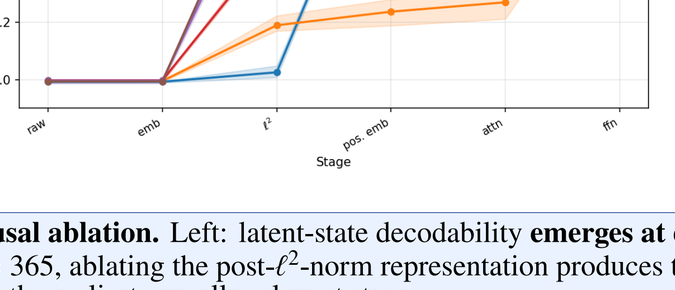
- MSE: Mean Squared Error on next-step squared returns. Noisy.
- NLL: Gaussian negative log-likelihood. It is uniquely minimized at $\mathbb{E}(\boldsymbol{r}_t^2 \mid \boldsymbol{h}_t) = \exp(\boldsymbol{h}_t)$:

$$\mathcal{L}_{\text{NLL}} = \frac{1}{N}\sum_{i=1}^{N}\left[\log\left(\widehat{r}^2_{i,T+1} + \delta\right) + \frac{r^2_{i,T+1}}{\widehat{r}^2_{i,T+1} + \delta}\right], \qquad \delta = 10^{-6}.$$

3. Two-Stage Latent-State Decomposition

Hypothesis I

1. **Latent volatility inference:** The model first maps past returns to hidden representations $\boldsymbol{z} = [z_1, \ldots, z_N] \in \mathbb{R}^{N \times d_{\text{model}}}$ that encode information about the next latent log-volatility state $\boldsymbol{h}_{T+1}$.
2. **Squared-return prediction:** the output head maps $\boldsymbol{z}$ to $\hat{\boldsymbol{r}}^2_{T+1}$.



Latent-state decodability vs. learned readout

- Latent-state decodability and output-head alignment across activation functions, model architectures, and training objectives. Entries report median [IQR] over $n = 10$ seeds.

Act.	Model	Loss	Ridge Probe R^2	Model Readout R^2
exp	Transformer	NLL	0.9069 [0.8827, 0.9175]	0.8975 [0.8733, 0.9065]
	Transformer	MSE	0.8794 [0.8699, 0.8929]	0.7539 [0.7151, 0.7846]
	MLP	NLL	0.8985 [0.8800, 0.9098]	0.8899 [0.8754, 0.9003]
	MLP	MSE	0.8034 [0.7789, 0.8332]	0.7543 [0.7232, 0.7975]
spl	Transformer	NLL	0.9007 [0.8844, 0.9122]	0.8191 [0.7783, 0.8407]
	Transformer	MSE	0.8810 [0.8684, 0.8963]	0.8011 [0.6929, 0.8119]
	MLP	NLL	0.8863 [0.8660, 0.9053]	0.7933 [0.7542, 0.8110]
	MLP	MSE	0.8211 [0.7811, 0.8436]	0.7207 [0.6032, 0.7473]

Output-head intervention under MSE + exponential head

- Positive gain indicates improvement. MAE improves consistently, while MSE effects are mixed.

		MSE			MAE		
Model	ω	Original	Probe Repair	Gain	Original	Probe Repair	Gain
Transformer	30	4.422	4.382	−1.39%	0.873	0.836	+5.28%
	178	7.470	7.469	−0.11%	0.987	0.958	+1.29%
	365	5.158	5.128	+0.47%	0.856	0.853	+0.66%
MLP	30	4.384	4.397	−1.34%	0.856	0.834	+3.89%
	178	7.433	7.446	−0.87%	0.994	0.954	+3.05%
	365	5.192	5.123	−0.01%	0.864	0.859	+1.18%

Interpretation

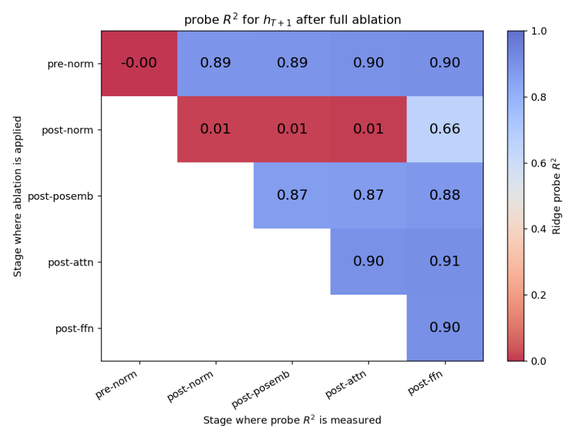
- Probe R^2 stays **robust** across architectures, losses, and output activations.
- Compared with MLP, **Transformer better preserves** the latent representation under MSE-induced noisy degradation.

Key takeaway: The network behaves like a **two-stage** latent-state pipeline: infer $\boldsymbol{h}_{T+1}$ first, then map it to squared-return forecasts. Under MSE, degradation can come from **readout misalignment**, not representation failure.

4. Stage-Wise Emergence of Latent-State Decodability

Hypothesis II

- **Stage-wise ridge probe R^2 for decoding $\boldsymbol{h}_{T+1}$:** We probe intermediate Transformer representations to locate the earliest stage where the next latent volatility state becomes linearly decodable.



Causal ablation. Left: latent-state decodability **emerges at different architectural stages** depending on ω. Right: for $\omega = 365$, ablating the post-ℓ^2-norm representation produces the strongest downstream collapse in decodability, identifying it as the earliest causally relevant stage.

Mechanistic reading

- **Long-cycle regimes** ($\omega = 178, 365, 730$): decodability appears immediately after ℓ^2 normalization.
- **Intermediate regimes** ($\omega = 30, 90$): the FFN contributes additional refinement.
- **Control regime** ($\omega = 21$): decodability remains near zero until positional embeddings are added.

MLP control

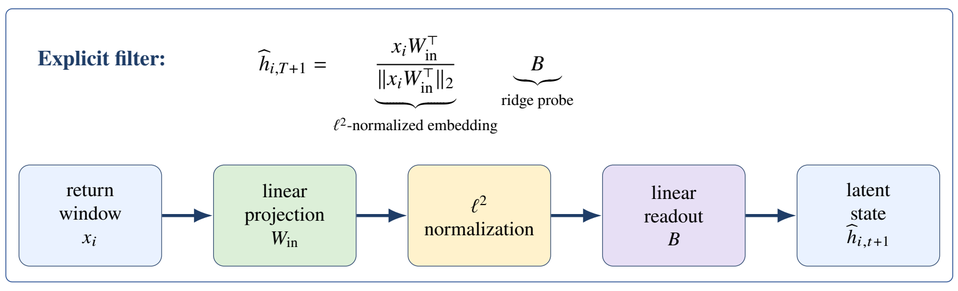
- The MLP exhibits a smoother and less localized rise in decodability across layers.
- The Transformer shows a sharper stage-specific pattern, supporting a localized computation.

Key takeaway: Transformers form latent-state information at **identifiable architectural stages** rather than diffusely throughout the network.

5. Explicit Latent-State Filter

Hypothesis III

- **For long-cycle dynamics ($\omega \geq 365$), the key computation is an explicit filter**: a learned linear projection of the return window, followed by ℓ^2 normalization, then a linear readout to the latent volatility state.



Ablation of filter components

Ridge R^2 for probing h_{t+1} under three ablation conditions (median [IQR], $n = 10$ seeds).

Condition	$\omega = 365$	$\omega = 730$
Linear → ℓ^2 norm → Linear	0.890 [0.866, 0.903]	0.921 [0.914, 0.933]
ℓ^2 norm on raw returns → Linear	−0.004 [-0.005, -0.002]	−0.003 [-0.004, -0.002]
Linear embedding only → Linear	−0.006 [-0.009, -0.003]	−0.005 [-0.015, -0.000]

- The learned projection and normalization are **jointly necessary**: neither raw normalization nor a linear embedding alone makes the latent state decodable.

Key takeaway: For long-cycle dynamics, the model has an **explicit** latent-state filter.

6. Takeaways & References

Poster takeaways

- **Two-stage latent-stage decomposition:** Latent-state information emerges at identifiable architectural stages, rather than diffusely throughout the network.
- **Understanding neural network's overfit to noise:** under MSE, degradation arises from readout misalignment rather than failure to encode the latent state.
- **Architecture matters:** the Transformer encodes the latent state more robustly than the MLP.
- **Knowledge discovery:** the model suggests an explicit latent-state filter in this controlled setting.

Selected references

1. Taylor, *Modelling Financial Time Series*, 1986.
2. Kim, Shephard, Chib, *Rev. Econ. Stud.*, 1998.
3. Harvey, Ruiz, Shephard, *Rev. Econ. Stud.*, 1994.
4. Vaswani et al., *NeurIPS*, 2017.
5. Alain and Bengio, arXiv:1610.01644, 2016.
6. Hewitt and Manning, *NAACL*, 2019.
7. Elhage et al., *Transformer Circuits Thread*, 2021.

Links & Resources

OpenReview · Code · Contact