Topics in Connectivity

Hernando Ombao

[a]King Abdullah University of Science and Technology
[b]University of Edinburgh

IMSI 2026 Chicago





Outline

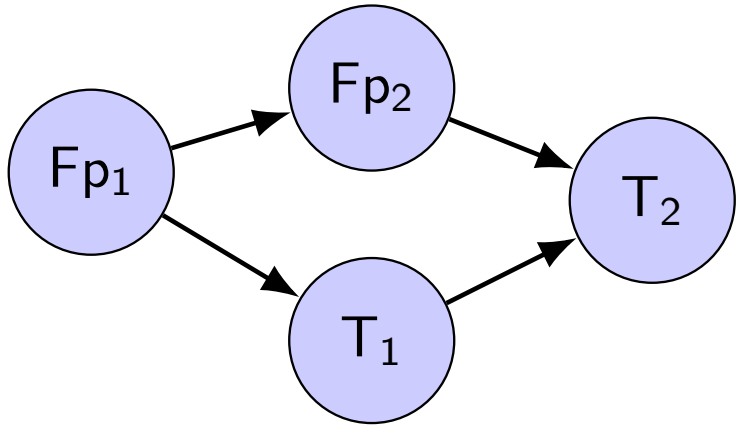
1. Deep learning / Granger Causality
   with Malik Shahid Sultan
2. Extremal Vulnerability
   with Paolo Redondo
3. Topological Effective Connectivity Modeling
   with Anass El-Yaagoubi

Outline

1. Deep learning / Granger Causality
   with Malik Shahid Sultan
2. Extremal Vulnerability
   with Paolo Redondo
3. Topological Effective Connectivity Modeling
   with Anass El-Yaagoubi

Why Study Granger Causality?

- Multivariate time series arise across domains:
  - **Neuroscience:** functional/effective connectivity between brain regions
  - **Finance:** lead–lag relationships between stocks
  - **Biology:** gene-regulatory pathways
- Goal: uncover the **cross-dependence structure** between components while also predicting future dynamics
- Physiological biomarkers (e.g., causal brain networks) can help differentiate patient subgroups



Unknown causal / connectivity structure

Granger Causality: Core Idea

Granger (1969)

Time series X_i **Granger-causes** X_j if the past of X_i helps predict the future of X_j (reduce uncertainty)*beyond* what is already explained by the past of all other series.

Two assumptions:

1. Cause happens *prior* to its effect
2. Cause has *unique* information about the future of its effect

Classical framework:

- **Full model:** predicts X_j using the past of *all* series
- **Reduced model:** predicts X_j using the past of all series *except* X_i
- Compare prediction errors of the two models

GC: Classic VAR

- $\mathbf{X}(t) = [X_1(t), \ldots, X_P(t)]'$
- VAR(1): $\mathbf{X}(t) = \Phi_1 \mathbf{X}(t-1) + \mathbf{E}(t)$
- In particular, for channel X_j,

$$X_j(t) = \sum_{p=1}^{P} \phi_{jp}^1 X_p(t-1) + E_j(t)$$

- $\mathbb{E}[X_j(t)|\mathbf{X}(t-1)] = \sum_{p=1}^{P} \phi_{jp}^1 X_p(t-1)$
- Granger Causality: $X_i \longrightarrow X_j$ if $\phi_{ji}^1 \neq 0$
- Applications in fMRI: Mixed Effects VAR (ME-VAR), NeuroImage, Gorrostieta et al.
- Later to generalize: $\mathbb{E}[X_j(t)|\mathbf{X}(\text{past})] = h_j(\mathbf{X}(\text{past}))$

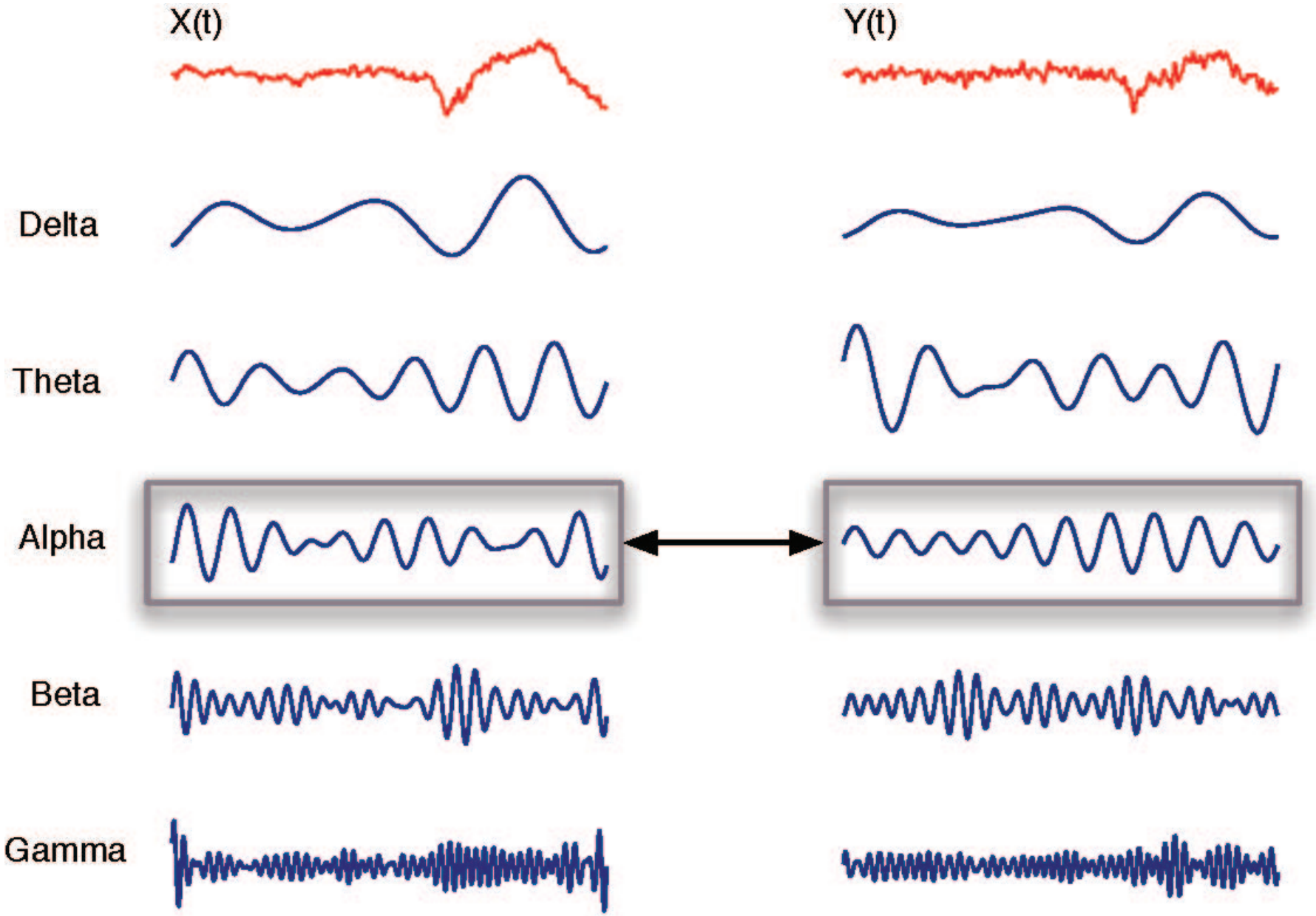
GC: Transfer Entropy $X_i \to X_j$

- Schreiber (2000)
- Compare two densities/models
  - Full model: $f^1 = f(X_i(\text{past}), X_j(t) | X_j(\text{past}))$
  - Null model: $f^0 = f(X_i(\text{past}) | X_j(\text{past})).f(X_j(t) | X_j(\text{past}))$
- Conditional independence: $X_i(\text{past}) \perp X_j(t) \mid X_j(\text{past})$
- Tranfer Entropy $X_i \to X_j$:

$$\begin{aligned} TE(X_j \to X_j) &= KL(f^1, f^0) \\ &= \text{Ent}[X_j \mid X_j(\text{past})] - \text{Ent}[X_j \mid X_j(\text{past}), X_i(\text{past})] \end{aligned}$$

- $TE \geq 0$, $TE(X_i \to X_j) = 0 \iff X_i \not\to X_j$
- $TE(X_i \to X_j) > 0$: Using the past of X_i (in addition to the past of X_j) reduces uncertainty in predicting the future of X_j.

Spectral Transfer Entropy



Spectral Transfer Entropy

- Redondo et al. (2025).
- General vs specific (spectral) dependence
  - $X_i \longleftrightarrow X_j$
  - $X_i^{\Omega_1}(t-\ell) \longrightarrow X_j^{\Omega_2}(t)$
- Compare two densities/models
  - Full model: $f^1 = f(X_i^{\Omega_1}(\text{past}), X_j^{\Omega_2}(t) | X_j^{\Omega_2}(\text{past}))$
  - Null model: $f^0 = f(X_i^{\Omega_1}(\text{past}) | X_j^{\Omega_2}(\text{past})) \; . \; f(X_j^{\Omega_2}(t) | X_j^{\Omega_2}(\text{past}))$
- Spectral Transfer Entropy

$$\text{STE}(X_i^{\Omega_1} \to X_j^{\Omega_2}) = KL(f^1, f^0)$$

- Estimation via D-vine copulas; Inference via resampling

From Linear VAR to Deep Learning

- **VAR models:** simple, interpretable, but cannot capture nonlinear dependence
- **Spectral Tranfer Entropy (STE):** based on joint/conditional densities, general, but computationally costly
- **Deep learning for GC:** exploits function-approximation power of DNNs
- Dominant paradigm (Tank et al., 2021): *component-wise* networks + **sparse regression**

$$X_p(t) = f_{p,\theta}\big(g_{p,\omega}(\mathbf{X}(t-1{:}t-K))\big) + E_p(t)$$

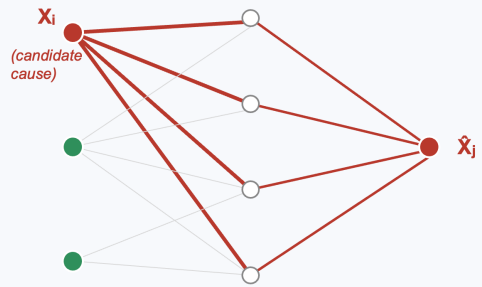
- Group Lasso applied to input layer $g_{p,\omega}$ to shrink weights $\rightarrow$ GC inferred from sparsity pattern
- **Limitation:** requires a *separate* network per component $\rightarrow$ computationally expensive as P grows (e.g., per-subject EEG analysis)

Component-wise Models Vs Joint Modeling

Why We Need a Mechanism to Discover $i \to j$ from a Jointly-Trained Network

A **Component-wise Network: GC is readable from weights**

Separate network $f_{j\theta}$ trained ONLY to predict X_j:

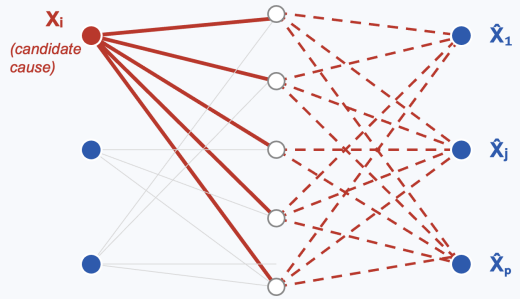


✓ GC($i \to j$) read directly from $\|\omega_i\|$, the input-layer weight norm for series i in $f_{j\theta}$

Straightforward , but needs P separate models, one per target series, plus sparsity tuning.

B **Jointly-Trained Network: GC is entangled in shared weights**

ONE network f_θ predicts ALL series jointly:



The weight on X_i feeds a SHARED hidden representation used by every output , its effect on X_j specifically cannot be isolated from ω alone

One model, no extra hyperparameters, but GC($i \to j$) is no longer legible from θ directly.

The Gap

?

Weights alone cannot reveal which series Granger-causes which once the model is trained jointly.

Our Contribution

A post-hoc mechanism for comparing full vs. reduced-input prediction squared error distributions under MC Dropout, that discovers the learned $i \to j$ structure without relying on input-layer sparsity.

The Central Question

Research Question

Can a **single jointly-trained** neural network, trained *only* for prediction (no sparsity-inducing penalty), still **learn** the true Granger causal structure?

- Prior view (Tank et al. 2021): weight-sharing in joint models makes GC inference from weights problematic
- **This paper's stance:** GC is fundamentally linked to *prediction*. A well-regularized joint model may implicitly encode GC structure, recoverable by comparing the **distribution of prediction errors** rather than by weights.

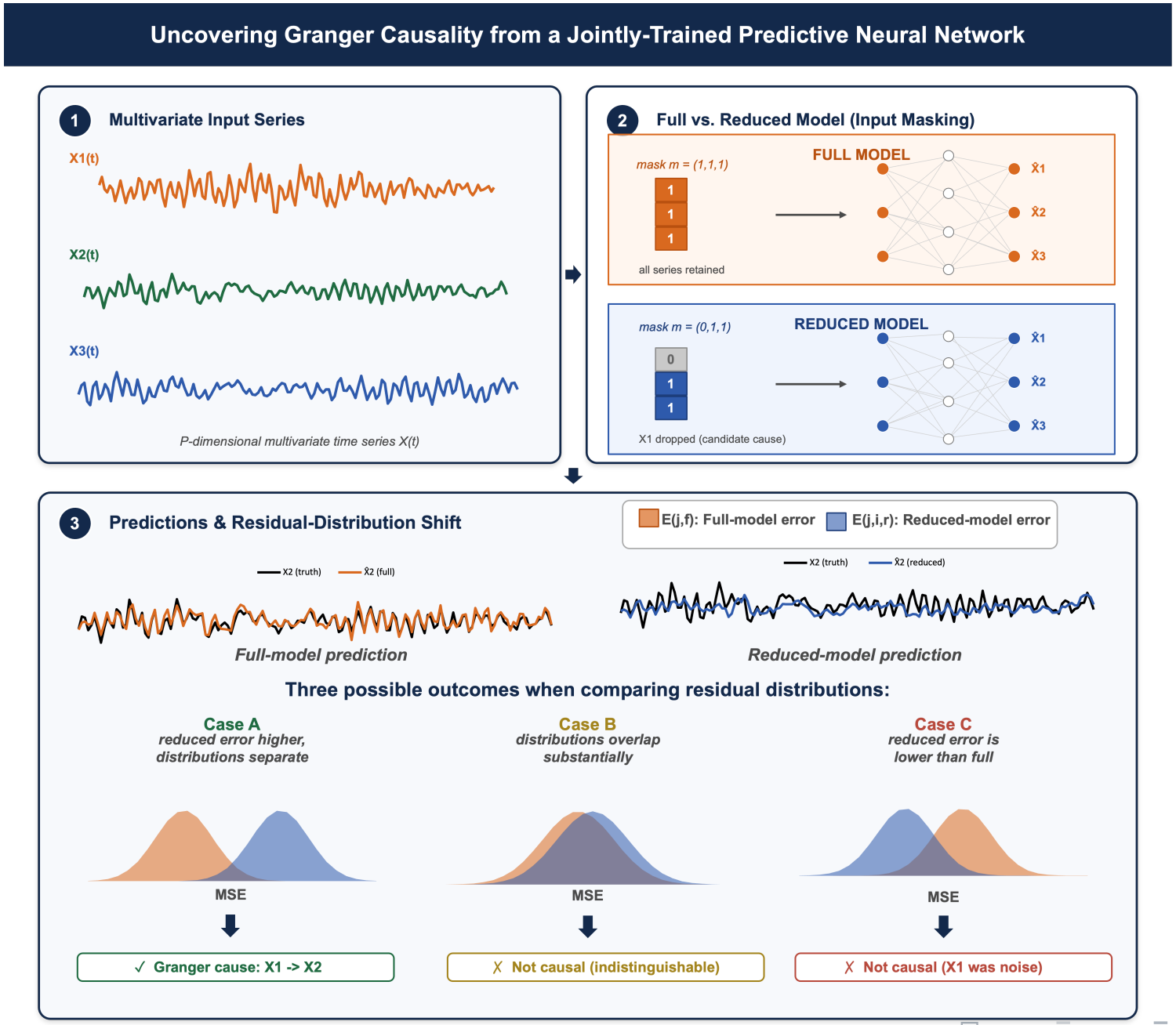
Key Contributions

1. **Neural Granger Causality Modeling with MC Dropout Inference:** We investigate whether neural networks jointly (rather than separately for each component) used to model multivariate time series, trained with dropout regularization, can learn the underlying Granger causal structure.

2. **Identification of GC without Input-Layer Sparsity:** We propose a novel approach to uncover the learned GC from a trained neural network without relying on sparse regression techniques or introducing additional tuneable hyperparameters.

3. **Architecture-wise Comparison:** We provide a comparative analysis of different neural architectures, Transformers [1], LSTMs [2], and CNNs [3], in terms of their capacity to identify Granger causality from time series data.

Overall Idea: Prediction → Structure



Problem Setup

Let $X(t) \in \mathbb{R}^P$ be a P-dimensional time series. Goal: estimate binary adjacency matrix A, where

$$X_i \to X_j \iff A[j, i] = 1.$$

Let f_θ be the neural network (parameters θ: weights + biases), trained via backpropagation.

Three training settings explored:

1. **No ILD:** dropout only in hidden layers
2. **ILD:** single forward pass, input-layer *and* hidden-layer dropout always active
3. **DP ILD (Dual Pass):** two forward passes per batch – with and without input-layer dropout. Hidden-layer dropout always active.

Full and Reduced Models

- P-dim multi-hot mask: $m = [m_1, \ldots, m_P]$
- $m_i = 0$: past of series X_i is dropped.
- The full model

$$\mathbf{X}_f(t) = f_\theta\big(\mathbf{X}(t{-}1{:}t{-}K)\big) + E_f(t)$$

- The reduced model

$$\mathbf{X}_r(t) = f_\theta\big(m * \mathbf{X}(t{-}1{:}t{-}K)\big) + E_r(t)$$

- Inference on GC conducted using estimated $\widehat{\theta}$:

$$\begin{aligned} \widehat{\mathbf{X}}_f(t) &= f_{\widehat{\theta}}\big(\mathbf{X}(t{-}1{:}t{-}K)\big) \\ \widehat{\mathbf{X}}_r(t) &= f_{\widehat{\theta}}\big(m * \mathbf{X}(t{-}1{:}t{-}K)\big) \end{aligned}$$

Full and Reduced Models

- $*$ = element-wise product
- Identical trained network $f_{\widehat{\theta}}$ used for both only the *input mask* changes
- MC Dropout allows *multiple stochastic forward passes* $\Rightarrow$ distributions of predictions, not single point estimates

MC Dropout for Uncertainty Quantification

Three MC-Dropout Training Settings for Neural Granger Causality

Dropout stays active at prediction time (Gal & Ghahramani, 2016); what differs is whether dropout is also applied to the INPUT layer

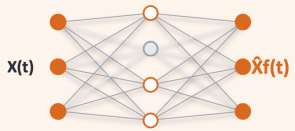
No ILD

No Input-Layer Dropout

Input-layer mask during training

1 1 1

m = (1,1,1) — always; no input dropout



Single forward pass · dropout only in hidden layers

$$\mathcal{L}_{\text{No ILD}}(\theta,\alpha) = \frac{1}{TP}\sum_{t=1}^{T}\sum_{i=1}^{P}\left(X_i(t) - \hat{X}_{if}(t)\right)^2$$

Plain MSE regression baseline. The network only ever sees the FULL input set, so a reduced model must be trained separately.

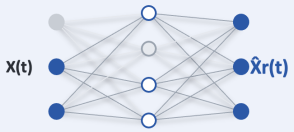
ILD

Input-Layer Dropout only

Input-layer mask during training

0 1 1

random mask only — no clean pass



Every mini-batch drops 1…P−1 series

$$\mathcal{L}_{\text{ILD}}(\theta,\alpha) = \frac{1}{TP}\sum_{t=1}^{T}\sum_{i=1}^{P}\left(X_i(t) - \hat{X}_{ir}(t)\right)^2$$

The no-dropout pass is removed. Tests whether random masking alone is enough to recover Granger-causal associations.

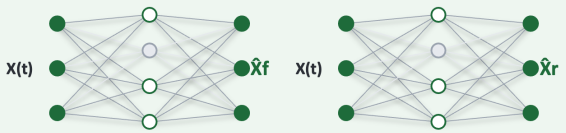
DP ILD

Dual-Pass Input-Layer Dropout

Input-layer mask during training

1 1 1 | 1 0 1

Pass 1 — full input | ***Pass 2 — random drop***



Both passes share the same weights ϑ

$$\mathcal{L}_{\text{DP ILD}}(\theta,\alpha) = \frac{1}{TP}\sum_{t=1}^{T}\sum_{i=1}^{P}\left(X_i(t) - \hat{X}_{if}(t)\right)^2 + \frac{1}{TP}\sum_{t=1}^{T}\sum_{i=1}^{P}\left(X_i(t) - \hat{X}_{ir}(t)\right)^2$$

Loss sums the full-input and the randomly-masked reconstruction. One network learns the full model AND many reduced models at once.

MC Dropout for Uncertainty Quantification

- Gal & Ghahramani (2016): keep dropout **active at test time** $\rightarrow$ ensemble of stochastic predictions $\rightarrow$ cheap uncertainty quantification
- Dropout probability α tuned via cross-validation

Three loss functions studied:

$$\mathcal{L}_{\text{NoILD}}(\theta, \alpha) = \frac{1}{TP} \sum_{t=1}^{T} \sum_{i=1}^{P} \left(\boldsymbol{x}_i(t) - \widehat{\boldsymbol{x}}_{if}(t) \right)^2 . \tag{1}$$

$$\mathcal{L}_{\text{ILD}}(\theta, \alpha) = \frac{1}{TP} \sum_{t=1}^{T} \sum_{i=1}^{P} \left(\boldsymbol{x}_i(t) - \widehat{\boldsymbol{x}}_{ir}(t) \right)^2 . \tag{2}$$

MC Dropout for Uncertainty Quantification

$$\mathcal{L}_{\text{DPILD}}(\theta, \alpha) = \frac{1}{TP} \sum_{t=1}^{T} \sum_{i=1}^{P} \left(\boldsymbol{x}_i(t) - \widehat{\boldsymbol{x}}_{if}(t) \right)^2 + \frac{1}{TP} \sum_{t=1}^{T} \sum_{i=1}^{P} \left(\boldsymbol{x}_i(t) - \widehat{\boldsymbol{x}}_{ir}(t) \right)^2 . \tag{3}$$

DP-ILD trains on *both* full and randomly-masked reduced views per batch (1 to $P-1$ series dropped); ILD uses *only* the masked view.

Algorithm 1: Obtaining Distributions of PSE



$S = \{s_1, \ldots, s_Q\}$



Q forward passes



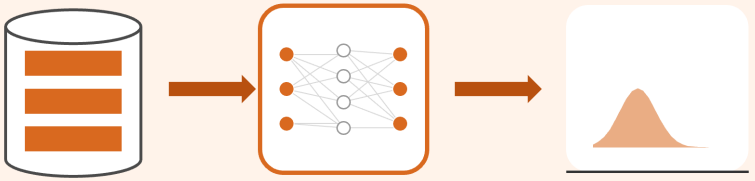
trained network

$f_{\hat{\theta}}$

test data $X(t-1{:}t-K)$

candidate cause i, target series j

1 Full Model Predictions



full input
(all series kept)

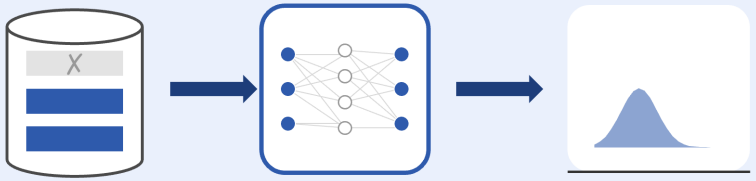
PSE distribution

for $q = 1, \ldots, Q$: set seed $s_q \rightarrow \hat{X}^q_{jf}(t) = f_{\hat{\theta}}(X(t-1{:}t-K)) \rightarrow$ compute PSE^q_{jf}

$$PSE_{jf} = \{PSE^1_{jf}, \ldots, PSE^Q_{jf}\}$$

Straightforward pass: nothing masked — the network sees everything.

2 Reduced Model Predictions



masked input
(series i dropped)

PSE distribution

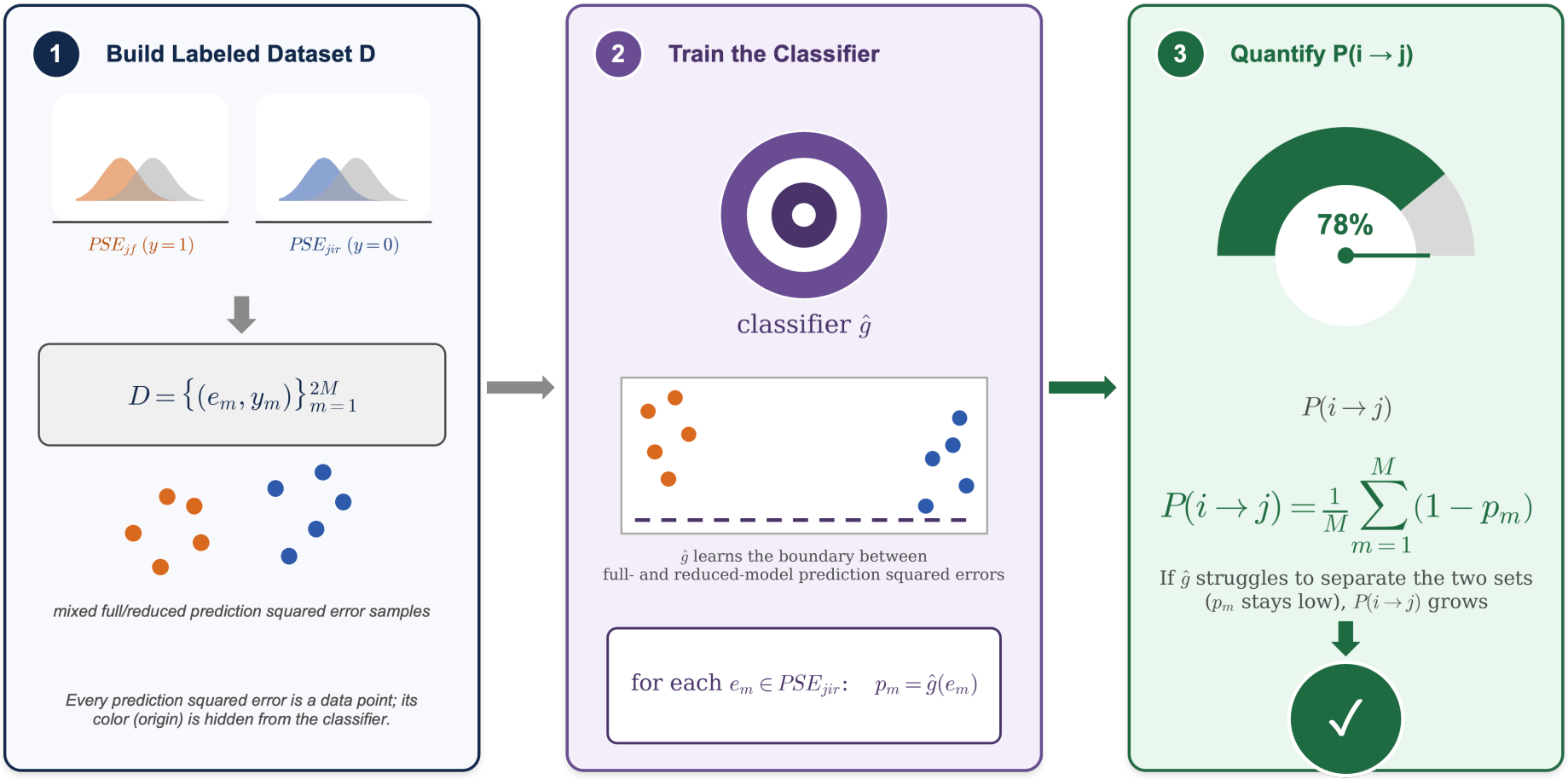
for $q = 1, \ldots, Q$: set seed s_q, mask $X_i \rightarrow \hat{X}^q_{jr}(t) = f_{\hat{\theta}}(X^{\prime}(t-1{:}t-K)) \rightarrow$ compute PSE^q_{jir}

$$PSE_{jir} = \{PSE^1_{jir}, \ldots, PSE^Q_{jir}\}$$

Same network, same seeds — only the mask on X_i changes.

Both loops share the same trained network and same seed sequence – only the input mask differs.

Algorithm 2: Detecting Granger Causality with Distribution Shift

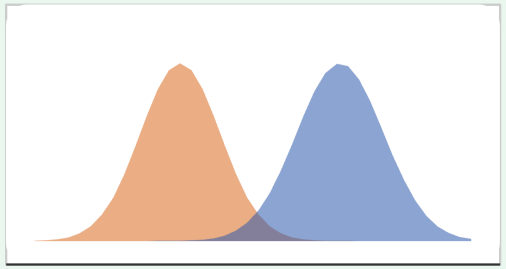


The classifier's inability to separate the two prediction squared error sets is itself the evidence, no manual threshold needed.

Interpreting the PSE Distributions: Three Possible Cases

Full model PSE_{jf}
Reduced model PSE_{jir}

A Clear Separation

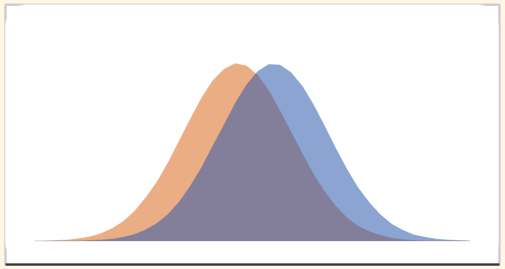


$\text{mean}(PSE_{jir}) > \text{mean}(PSE_{jf})$, distributions separate

$P(i \to j)$ computed from Algorithm 2 $\to$ high

✓ **Granger cause confirmed: i → j**

B Substantial Overlap

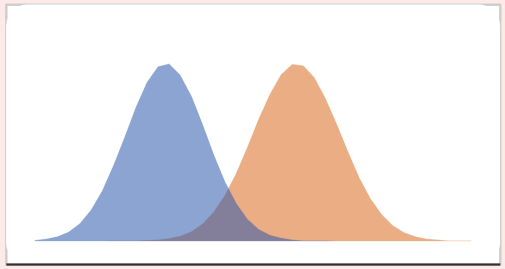


PSE_{jf} and PSE_{jir} overlap substantially

$P(i \to j) \approx 0.5$ (uninformative)

✗ **Not causal, too ambiguous to decide**

C Reduced Error Is Lower



$\text{mean}(PSE_{jir}) < \text{mean}(PSE_{jf})$

$P(i \to j) \equiv 0$

Not causal , X_i was adding noise, not signal

$$P(i \to j) = \frac{1}{M} \sum_{m=1}^{M} (1 - p_m) \quad \text{if } \text{mean}(PSE_{jir}) \geq \text{mean}(PSE_{jf}), \quad \text{else } 0$$

Simulation Design Overview

- **Benchmarks tested:**
  - Chain / Fork / Collider (3-variable canonical causal structures)
  - VAR(2) with varying sparsity
  - Lorenz-96 (20-dimensional chaotic system)
  - Netsim (realistic fMRI BOLD simulations, data-scarce regime)
- **Architectures:** LSTM, CNN, Transformer, each under No-ILD / DP-ILD / ILD
- **Baselines:** cLSTM (Tank et al. 2021), GVAR (Marcinkevičs & Vogt 2021), CUTS (Cheng et al. 2023)
- **Metrics:** AUCROC and AUPRC against ground-truth adjacency

Setup and notation

At each time t we feed the network the **past K lags** $\boldsymbol{X}_{t-K:t-1} = (\boldsymbol{X}_{t-K}, \ldots, \boldsymbol{X}_{t-1})$, $\boldsymbol{X}_s \in \mathbb{R}^P$, and fit a parametric map $\hat{\boldsymbol{X}}_t = f_\theta(\boldsymbol{X}_{t-K:t-1})$.

Symbol	Meaning	Value
B	batch size	64
K	sequence length (number of past lags)	data dependent
P	time series components (input channels)	data dependent
H	hidden units / channels	32
L	recurrent / conv. layers	2
M	MLP head layers	2
A	attention heads	2
F	filter size	3

A mini-batch is a tensor $\mathrm{X} \in \mathbb{R}^{B \times K \times P}$: B windows, each holding K lags of all P series. Every layer is a map between such tensors.

LSTM: a nonlinear state-space model

A linear Gaussian state-space model is

$$\boldsymbol{h}_k = \boldsymbol{A}\boldsymbol{h}_{k-1} + \boldsymbol{B}\boldsymbol{x}_k + \varepsilon_k, \qquad y_k = \boldsymbol{C}\boldsymbol{h}_k + \eta_k.$$

The LSTM replaces the linear transition by a *gated* nonlinear one, and adds a second state $\boldsymbol{c}_k$ (the "cell") carrying long-range information additively.

Two states propagate across the K lags: $\boldsymbol{h}_k \in \mathbb{R}^H$ (output) and $\boldsymbol{c}_k \in \mathbb{R}^H$ (memory), $k = 1, \ldots, K$.

LSTM: the cell equations

Indexing the K lags in the window by $k = 1, \ldots, K$ (so $\boldsymbol{x}_k = \boldsymbol{X}_{t-K+k-1} \in \mathbb{R}^P$), with $\sigma(u) = 1/(1+e^{-u})$ elementwise:

$$
\begin{aligned}
\boldsymbol{f}_k &= \sigma(\boldsymbol{W}_f \boldsymbol{x}_k + \boldsymbol{U}_f \boldsymbol{h}_{k-1} + \boldsymbol{b}_f) && \text{forget gate} \\
\boldsymbol{i}_k &= \sigma(\boldsymbol{W}_i \boldsymbol{x}_k + \boldsymbol{U}_i \boldsymbol{h}_{k-1} + \boldsymbol{b}_i) && \text{input gate} \\
\boldsymbol{o}_k &= \sigma(\boldsymbol{W}_o \boldsymbol{x}_k + \boldsymbol{U}_o \boldsymbol{h}_{k-1} + \boldsymbol{b}_o) && \text{output gate} \\
\tilde{\boldsymbol{c}}_k &= \tanh(\boldsymbol{W}_c \boldsymbol{x}_k + \boldsymbol{U}_c \boldsymbol{h}_{k-1} + \boldsymbol{b}_c) && \text{candidate} \\
\boldsymbol{c}_k &= \boldsymbol{f}_k \odot \boldsymbol{c}_{k-1} + \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k && \text{state update} \\
\boldsymbol{h}_k &= \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k) && \text{output}
\end{aligned}
$$

with $\odot$ the Hadamard product, $\boldsymbol{W}_\bullet \in \mathbb{R}^{H \times P}$, $\boldsymbol{U}_\bullet \in \mathbb{R}^{H \times H}$ and $\boldsymbol{h}_0 = \boldsymbol{c}_0 = \boldsymbol{0}$.

Read $\boldsymbol{c}_k$ as an adaptive exponentially weighted moving average over the K lags: $\boldsymbol{f}_k$ is a learned, lag-varying discount factor.

LSTM: cell diagram

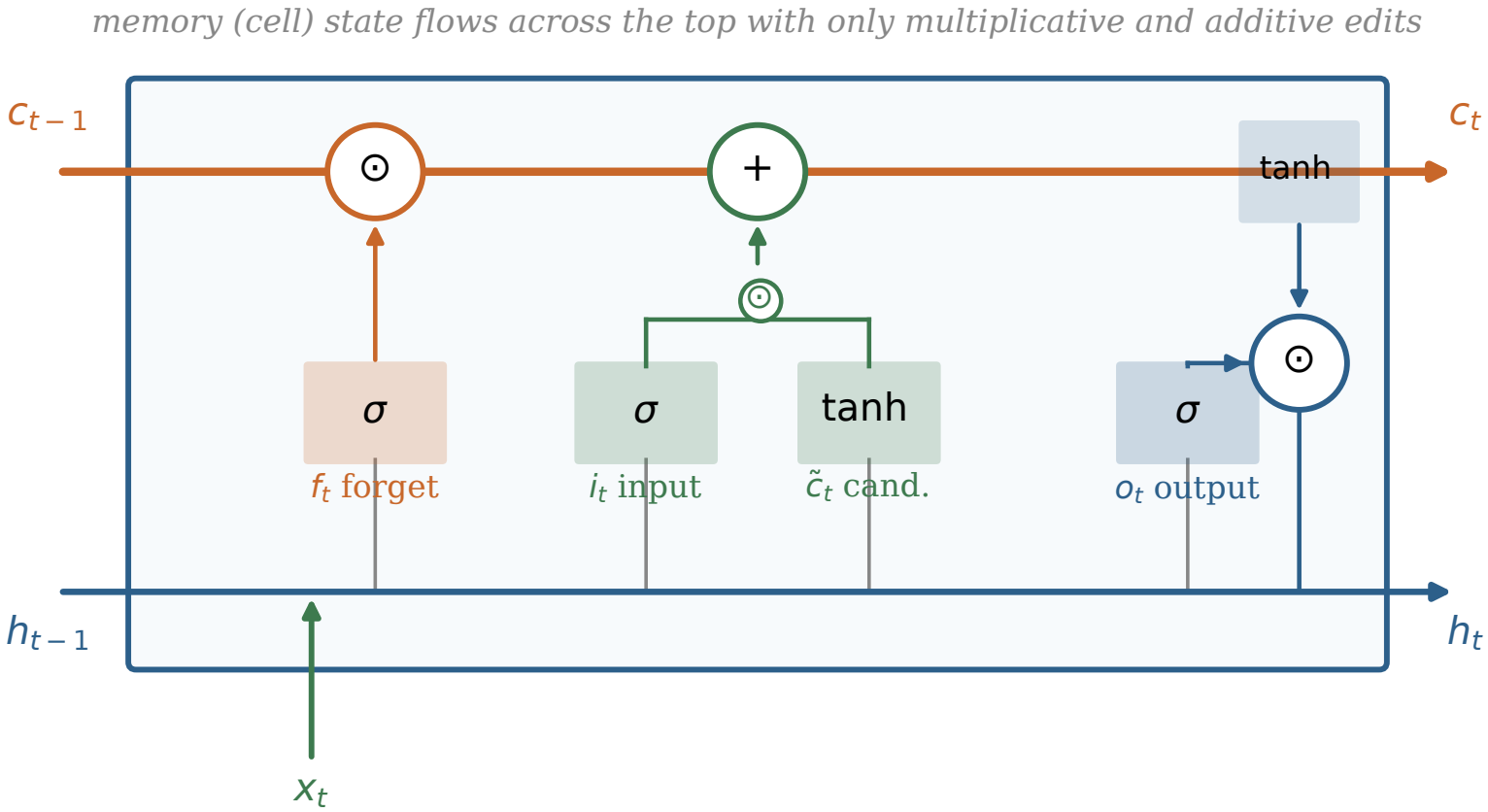


Figure: One LSTM cell: gates $\boldsymbol{f}_k, \boldsymbol{i}_k, \boldsymbol{o}_k$ controlling flow into and out of the memory state $\boldsymbol{c}_k$.

LSTM: unrolling over the K lags

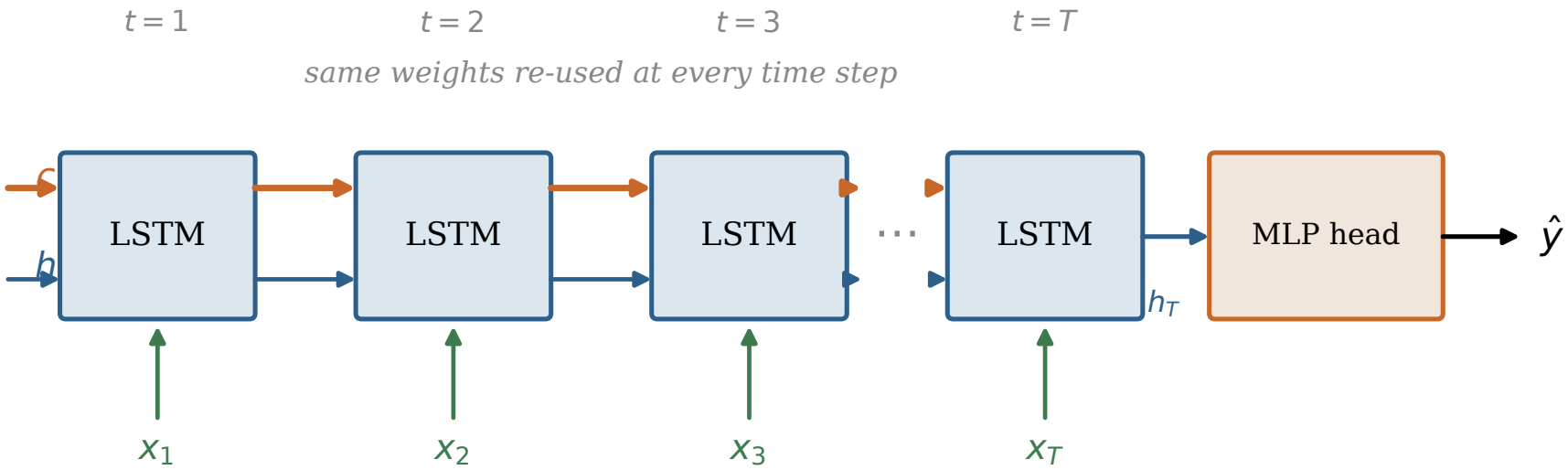


Figure: The same cell applied at each of the K lags in the window (weights shared across k); the horizontal line carrying $\boldsymbol{c}_k$ is the memory highway.

CNN: learned moving-average filters

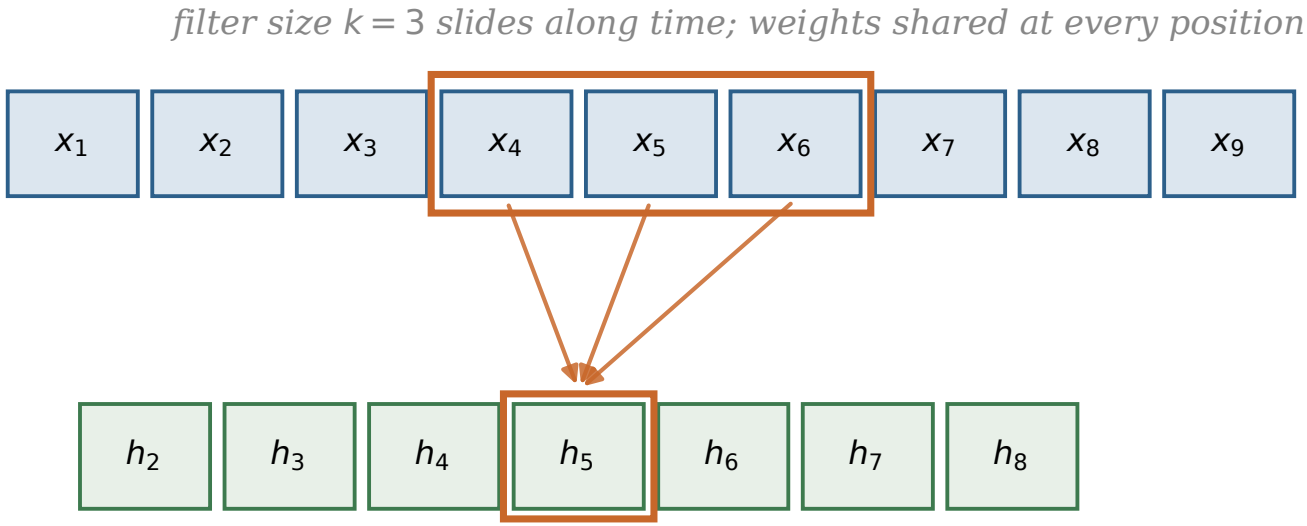
A 1-D convolution with $F = 3$ runs along the lag axis and computes, for output channel c,

$$h_k^{(c)} = \phi\left(\sum_{j=1}^{P}\sum_{s=-1}^{1} w_{j,s}^{(c)}\, x_{k+s}^{(j)} + b^{(c)}\right), \qquad \phi = \mathrm{ReLU},$$

where j indexes the P series and k the lag position, $k = 1, \ldots, K$.
Statistical reading. A local linear filter , a moving average of window 3 over the lags , followed by a nonlinearity, with $H = 32$ filters *learned* rather than chosen a priori.

- **Weight sharing**: same $w^{(c)}$ at every lag $k \Rightarrow$ translation equivariance, few parameters.
- **Locality**: h_k depends only on a window of lags around k; stacking $L = 2$ layers widens the receptive field to 5 lags.

CNN: the convolution



$$h_t = \phi(w_{-1}x_{t-1} + w_0 x_t + w_{+1}x_{t+1} + b)$$

Figure: A size-3 filter sliding along the lag axis over the P input channels, producing one output channel; $H = 32$ such filters run in parallel.

Transformer: the attention computation

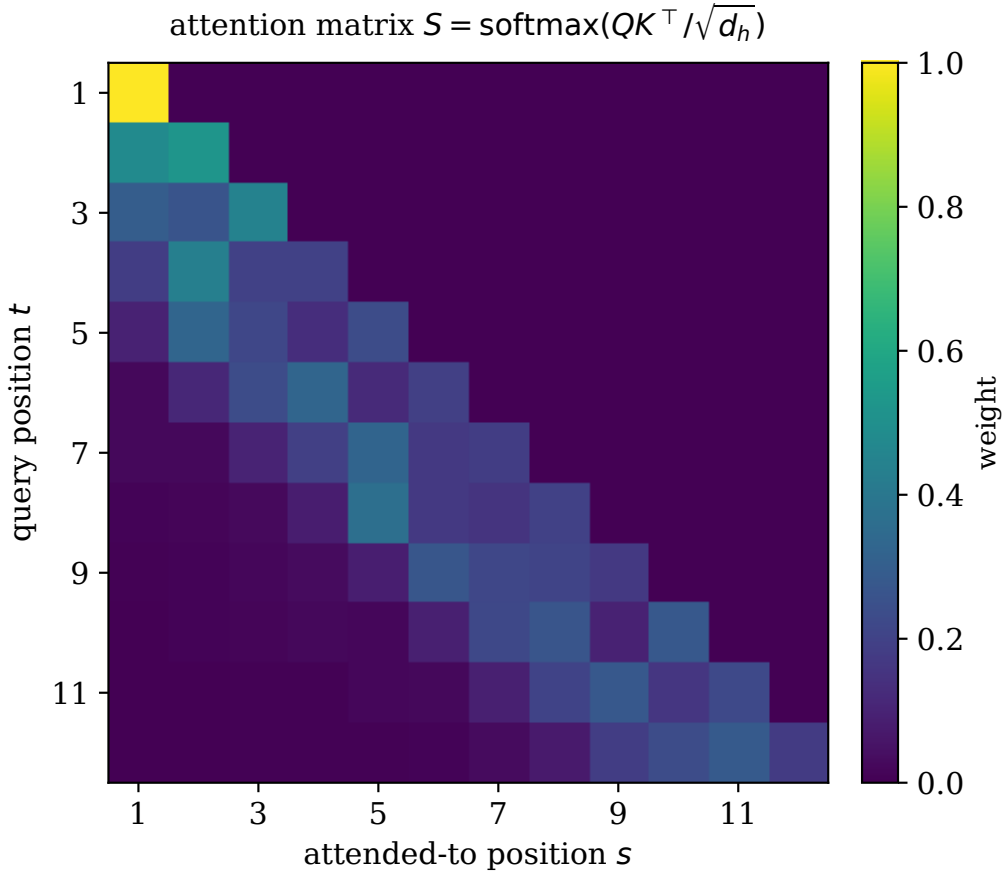
Stack the window $\mathrm{X} \in \mathbb{R}^{K \times P}$ and set $\boldsymbol{Q} = \mathrm{X}\boldsymbol{W}_Q$, $\boldsymbol{\mathcal{K}} = \mathrm{X}\boldsymbol{W}_{\mathcal{K}}$, $\boldsymbol{V} = \mathrm{X}\boldsymbol{W}_V$ with $\boldsymbol{W}_\bullet \in \mathbb{R}^{P \times d_h}$. Then

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{\mathcal{K}}, \boldsymbol{V}) = \underbrace{\mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{\mathcal{K}}^\top}{\sqrt{d_h}}\right)}_{\boldsymbol{S} \in \mathbb{R}^{K \times K},\ \text{rows sum to 1}} \boldsymbol{V}.$$

$\boldsymbol{S}$ is **row-stochastic**: row k gives the weights placed on all K lags when forming the representation at lag k. Every output sees the whole window, so the receptive field is the full K lags.

With $A = 2$ heads and width $H = 32$, each head uses $d_h = H/A = 16$; head outputs are concatenated and mixed by $\boldsymbol{W}_O \in \mathbb{R}^{32 \times 32}$.

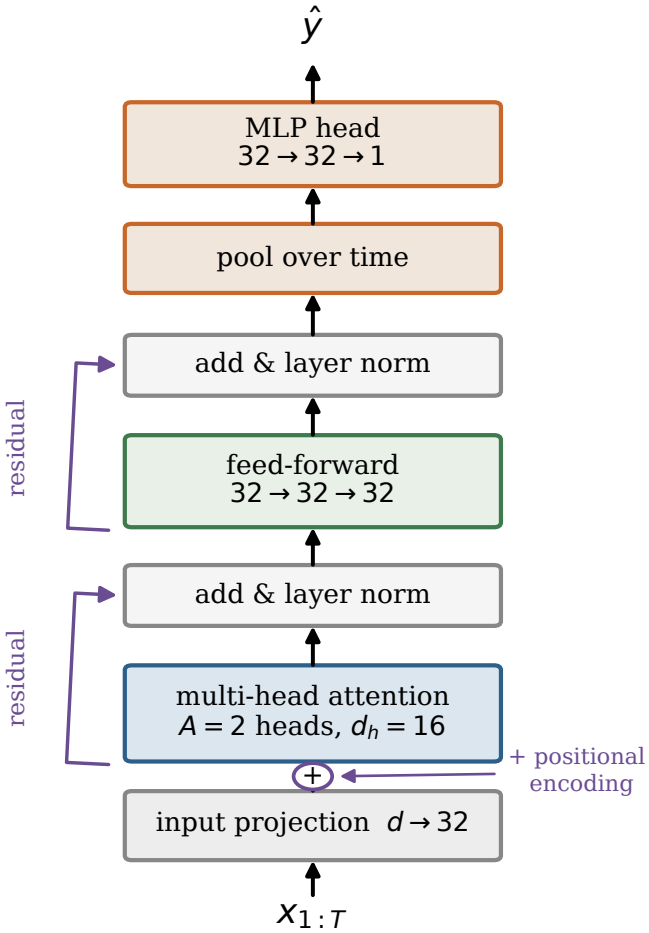
Transformer: attention weights are interpretable



each row sums to 1: a data-dependent smoothing kernel

Figure: Attention matrix $\boldsymbol{S} \in \mathbb{R}^{K \times K}$: entry (k, s) is the weight lag k places on lag s. Rows sum to one.

Transformer: the full block

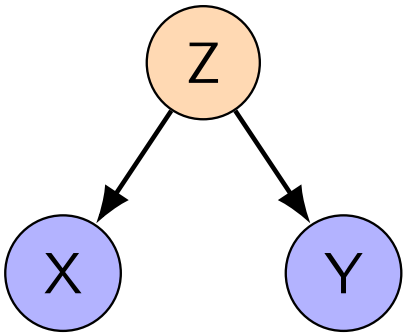


Multi-head attention $\to$ residual + layer norm $\to$ position-wise feed-forward $\to$ residual + layer norm.

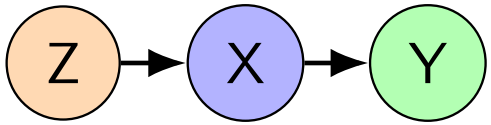
Attention is permutation-equivariant, so a **positional encoding** is added to $\boldsymbol{x}_k$ before the block; without it the ordering of the K lags is unrecoverable.

Residual connections keep the gradient path short, which is what makes depth trainable.

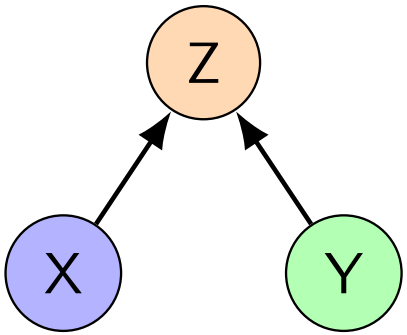
Canonical Causal Structures



Fork



Chain



Collider

- **Fork:** past of Z causes future of both X and Y
- **Chain:** past of Z causes X, past of X causes Y
- **Collider:** past of X and Y jointly cause Z

Data Generating Process

Each series follows a nonlinear autoregressive process:

$$X_t(i) = \sum_{k=1}^{K} \phi_k^{(i)} X_{t-k}(i) + \sum_{j \in \mathrm{Pa}(i)} f_{ij}\left(X_{t-1}(j)\right) + \varepsilon_t(i)$$

- AR order $K = 2$; coefficients $\phi_k^{(i)} = 0.5^k$
- $\mathrm{Pa}(i)$ = parent (causal) variables of i
- Nonlinear influence function:

$$f_{ij}(X_{t-1}(j)) = \tanh(|X_{t-1}(j)|) + \sin(|X_{t-1}(j)|)$$

- Noise $\varepsilon_t(i) \sim \mathcal{N}(0, \sigma_e^2)$, $\sigma_e^2 = 0.1$

Results: Chain, Fork, and Collider Structures

Chain Structure

AUCROC

	LSTM	CNN	Trans.
No ILD	1.00±.00	1.00±.00	1.00±.00
DP ILD	0.90±.16	0.98±.05	0.95±.10
ILD	0.60±.29	0.60±.29	0.61±.28

AUPRC

	LSTM	CNN	Trans.
No ILD	1.00±.00	1.00±.00	1.00±.00
DP ILD	0.91±.14	0.97±.07	0.95±.10
ILD	0.60±.28	0.60±.28	0.62±.26

Fork Structure

AUCROC

	LSTM	CNN	Trans.
No ILD	1.00±.00	1.00±.00	1.00±.00
DP ILD	1.00±.00	1.00±.00	1.00±.00
ILD	0.86±.19	0.84±.24	0.89±.14

AUPRC

	LSTM	CNN	Trans.
No ILD	1.00±.00	1.00±.00	1.00±.00
DP ILD	1.00±.00	1.00±.00	1.00±.00
ILD	0.82±.26	0.80±.29	0.87±.17

Collider Structure

AUCROC

	LSTM	CNN	Trans.
No ILD	1.00±.00	1.00±.00	1.00±.00
DP ILD	0.90±.14	0.88±.18	1.00±.00
ILD	0.58±.35	0.70±.34	0.71±.23

AUPRC

	LSTM	CNN	Trans.
No ILD	1.00±.00	1.00±.00	1.00±.00
DP ILD	0.87±.19	0.86±.20	1.00±.00
ILD	0.58±.34	0.67±.37	0.70±.25

- Values are mean ± standard deviation across 5 random datasets. Trans. = Transformer.
- **No ILD** ≈ perfect everywhere; **ILD** degrades all structures/architectures; **DP ILD** sits in between, matching No-ILD on Fork.

Interpretation: Chain, Fork, and Collider Results

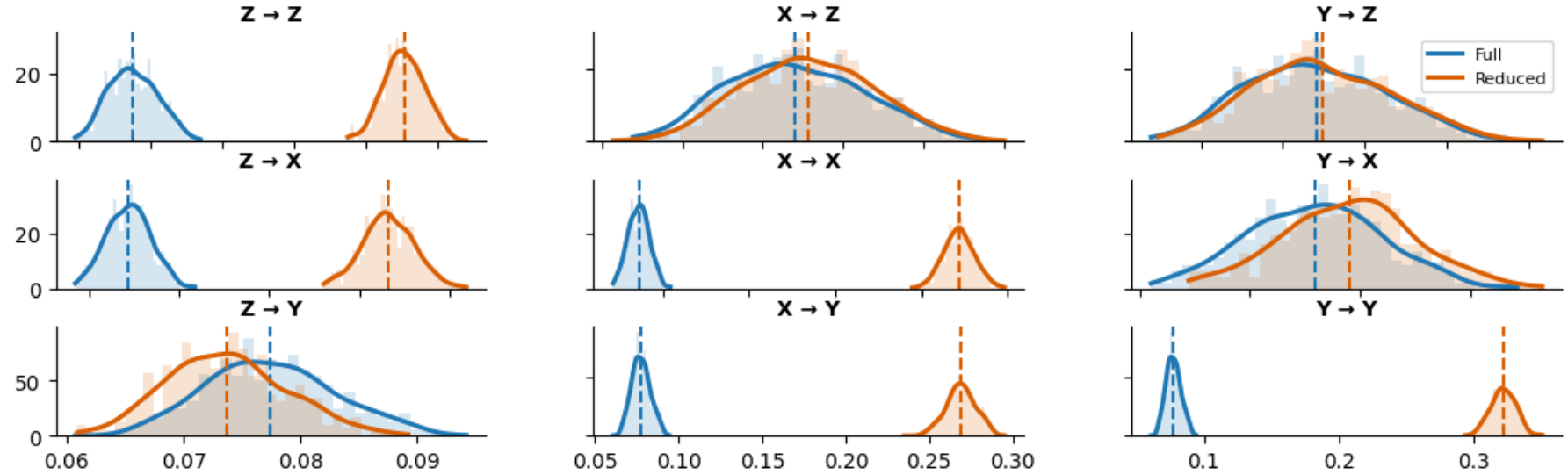
- **No-ILD training is essentially perfect** across all three canonical structures, all three architectures, and both metrics (AUCROC/AUPRC ≈ 1.00) confirming that unrestricted joint prediction reliably recovers simple 3-node causal structures.
- **DP-ILD closely tracks No-ILD**, and is indistinguishable from it on the **Fork** structure (perfect scores under both regimes). Degradation appears only under Chain and Collider, and only mildly (AUCROC ≥ 0.88 in nearly all cases).
- **ILD causes the sharpest degradation** in every structure and every architecture, AUCROC drops as low as 0.58–0.61 for Chain and Collider. This confirms that training *exclusively* on masked input degrades the model's ability to learn true Granger-causal structure.

- **Structure-dependent robustness to ILD:** Fork is the most forgiving structure under ILD (AUCROC 0.84–0.89), while Collider is the most fragile (AUCROC as low as 0.58 for LSTM). Chain sits in between.
- **Transformer is the most robust architecture** under degraded training regimes, achieving the highest or joint-highest AUCROC/AUPRC under ILD in 5 of 6 structure/metric combinations.

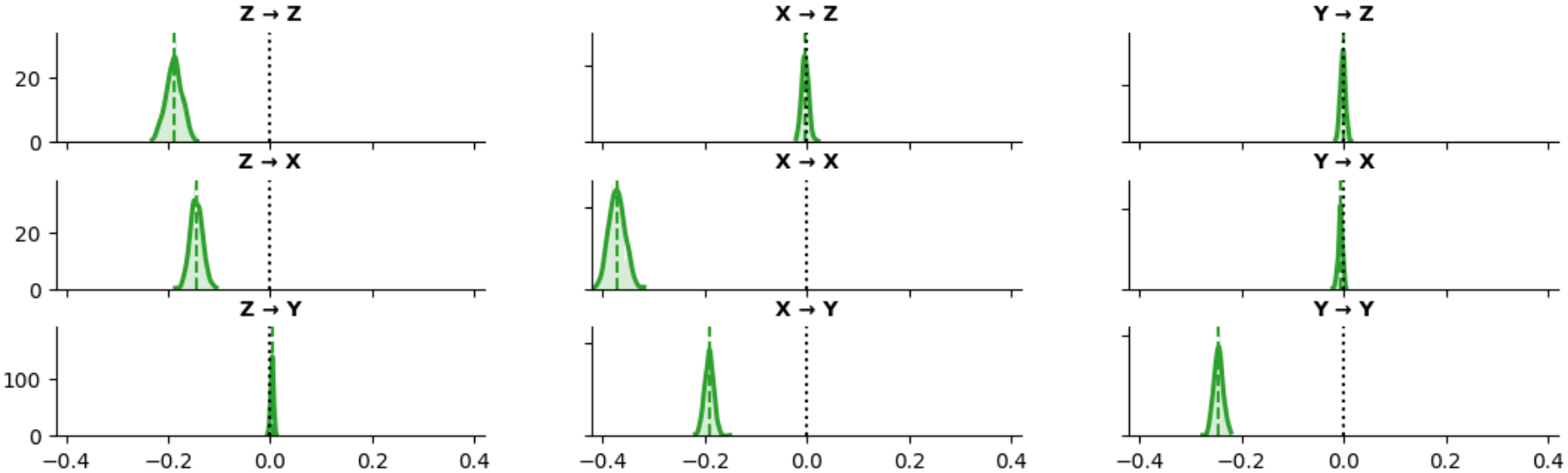
Takeaway

Input-layer dropout during training consistently harms Granger-causal recovery regardless of architecture or causal structure; dual-pass training (DP-ILD) largely mitigates this cost.

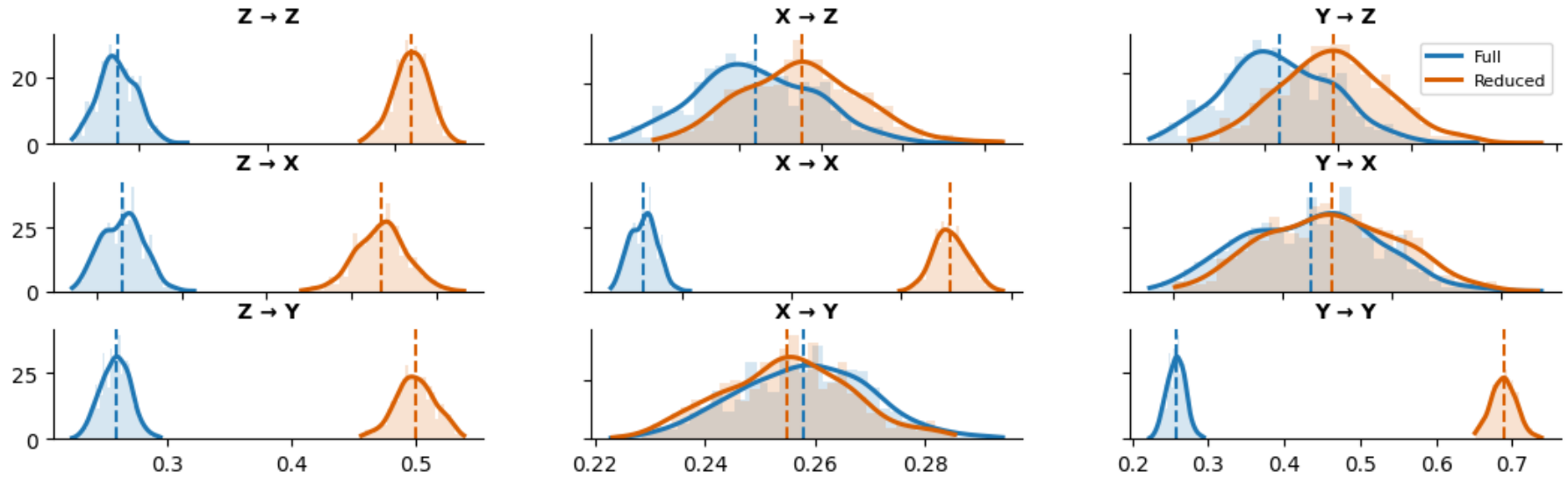
A **Distribution of Full and Reduced Model PSE**



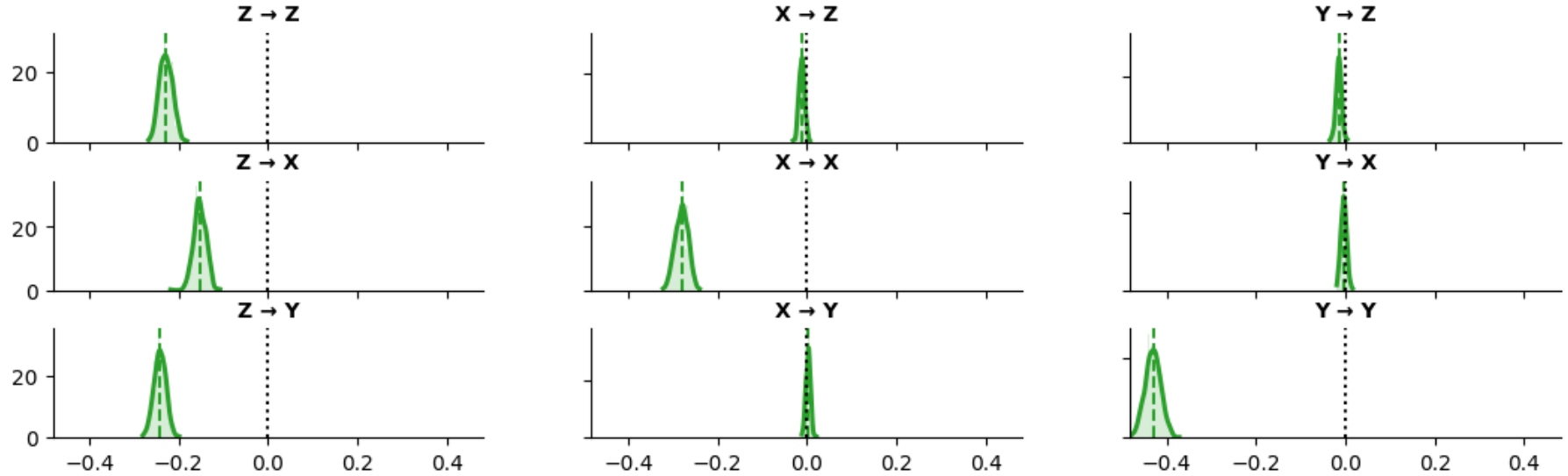
B **Distribution of ΔPSE (Full − Reduced)**



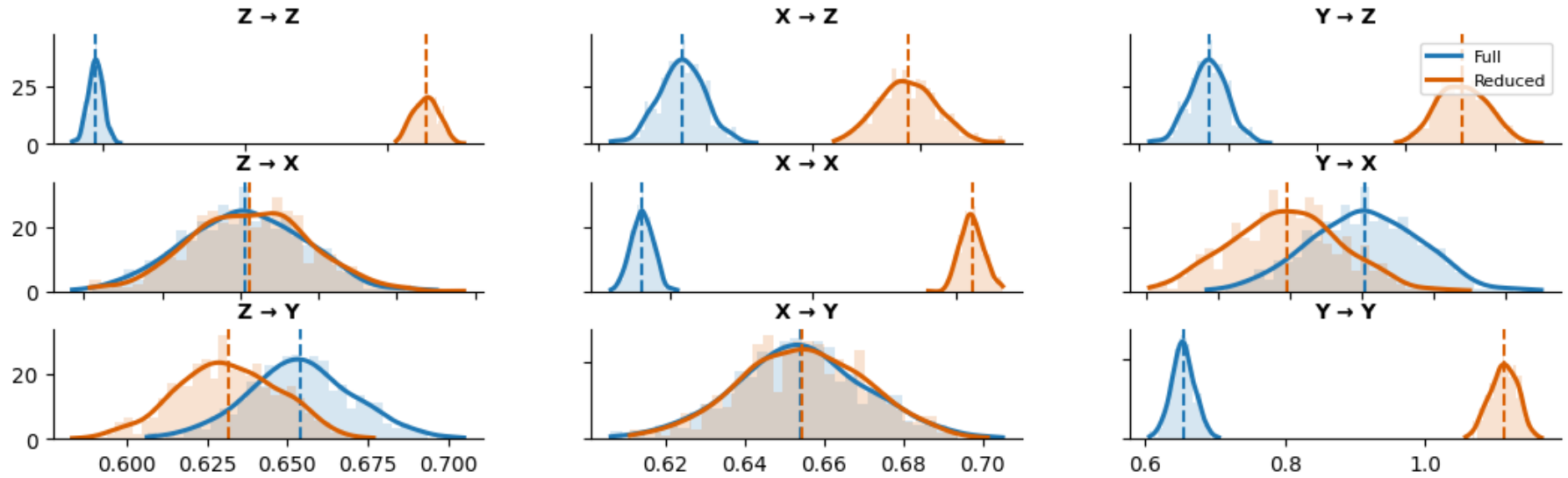
A **Distribution of Full and Reduced Model PSE**



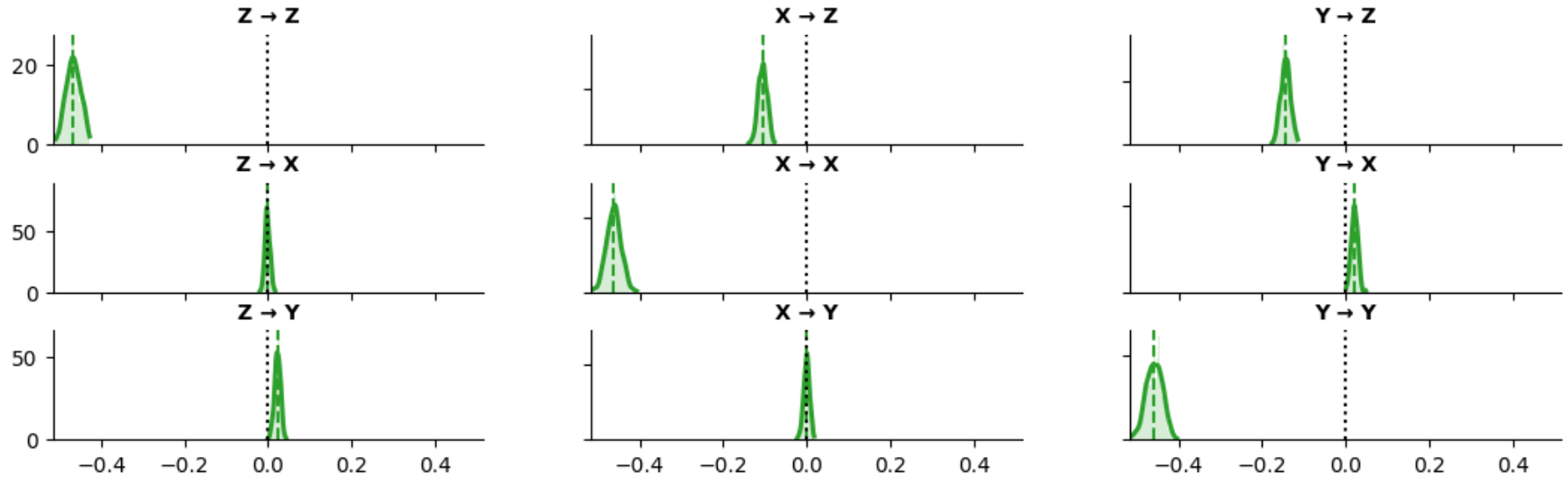
B **Distribution of ΔPSE (Full − Reduced)**



A **Distribution of Full and Reduced Model PSE**



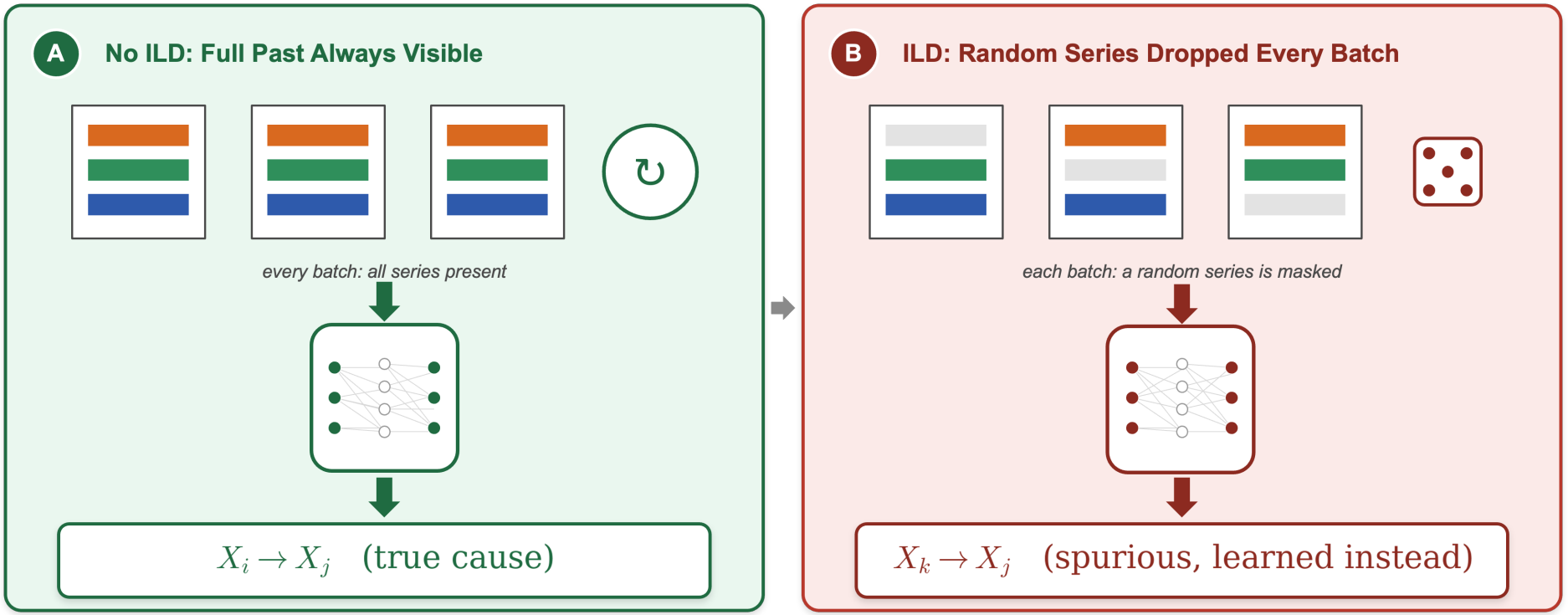
B **Distribution of ΔPSE (Full − Reduced)**



Why Does Input-Layer Dropout Hurt?

- When the true cause is dropped during training, the network learns to **lean on spuriously correlated** series instead
- This teaches the network mappings that don't reflect the *true* GC structure
- DP-ILD (dual pass) is less damaging than pure ILD, but adds computational overhead and still underperforms No-ILD

Why Does Input-Layer Dropout Hurt?



Model forced to substitute a correlated but non-causal series to compensate

When the true cause is dropped during training, the network learns to lean on spuriously correlated series instead teaching it mappings that don't reflect the true GC structure.

DP-ILD (dual pass) is less damaging than pure ILD, but adds computational overhead and still underperforms No-ILD.

VAR Performance Comparison

Model – Regime	AUCROC	AUPRC
LSTM – No ILD	1.00 ± 0.00	1.00 ± 0.00
LSTM – DP ILD	0.98 ± 0.04	0.96 ± 0.05
LSTM – ILD	0.80 ± 0.06	0.62 ± 0.04
CNN – No ILD	1.00 ± 0.00	1.00 ± 0.00
CNN – DP ILD	0.98 ± 0.03	0.92 ± 0.11
CNN – ILD	0.88 ± 0.10	0.70 ± 0.19
Transformer – No ILD	1.00 ± 0.00	1.00 ± 0.00
Transformer – DP ILD	1.00 ± 0.00	0.99 ± 0.01
Transformer – ILD	0.70 ± 0.23	0.52 ± 0.36

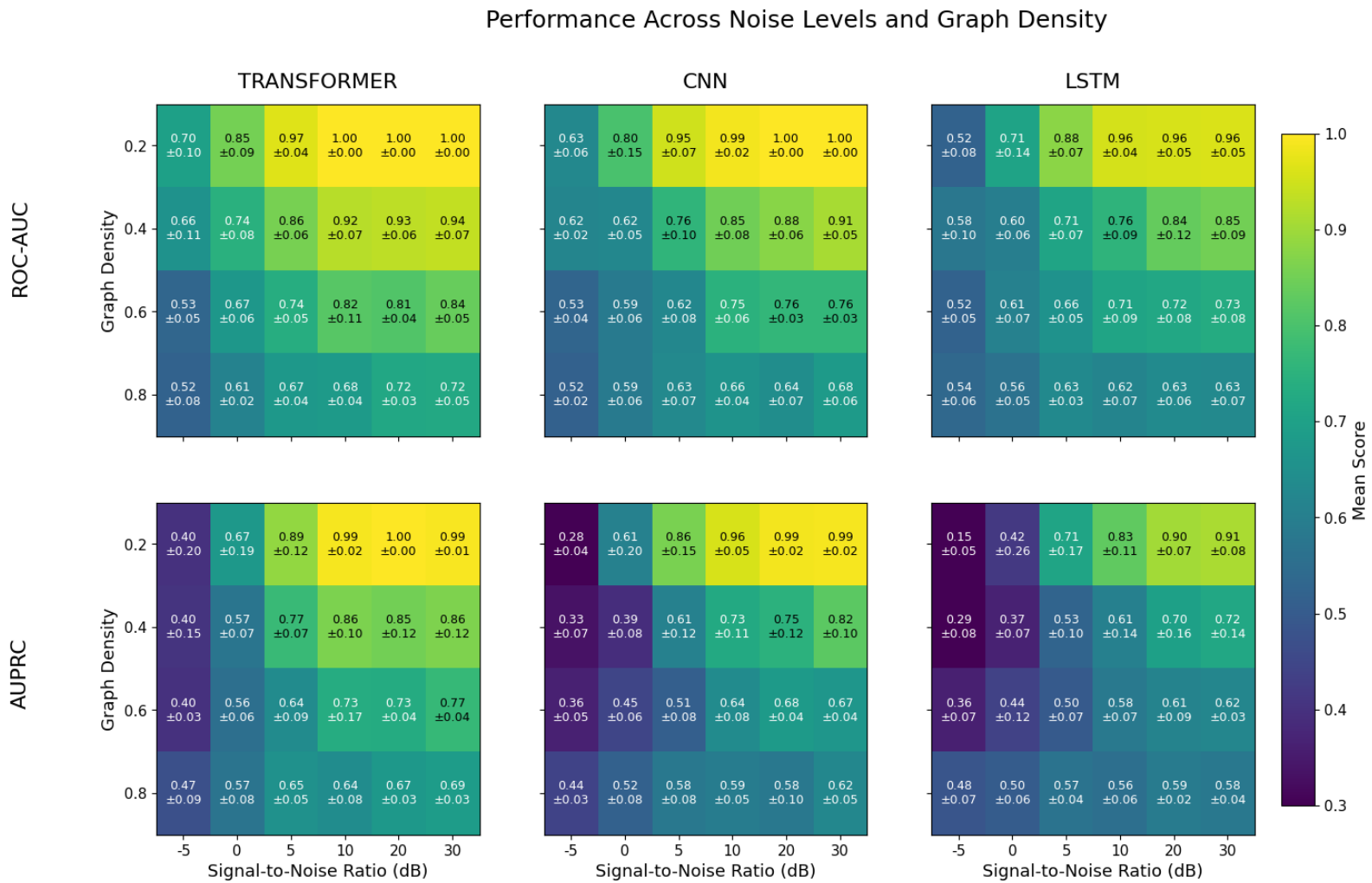
Sparse-Regression Baselines

Model	AUCROC	AUPRC
GVAR	0.98 ± 0.01	0.92 ± 0.02
CUTS	1.00 ± 0.00	1.00 ± 0.00
cLSTM	0.98 ± 0.01	0.83 ± 0.07

Key Finding

Unrestricted No-ILD models **match or beat** purpose-built sparse-regression baselines, without tuning extra sparsity hyperparameters.

VAR(2): Effect of Sparsity Level



Sensitivity Analysis: Effect of Graph Density

Setup: VAR(2) model, $P = 10$ channels, $T = 800$, graph density $gd \in [0, 1]$ (fraction $P \times gd$ of causal parents per channel), varying Gaussian noise levels, all models trained under **No-ILD**.

- **Transformer performs best when the ground truth is sparse and SNR is high**,its advantage over CNN and LSTM is most pronounced in this regime.
- **AUPRC declines more steeply than AUROC** as *gd* increases and SNR decreases for *all* models, precision-recall performance is far more sensitive to graph density and noise than AUROC suggests.

Two possible explanations for the decline:

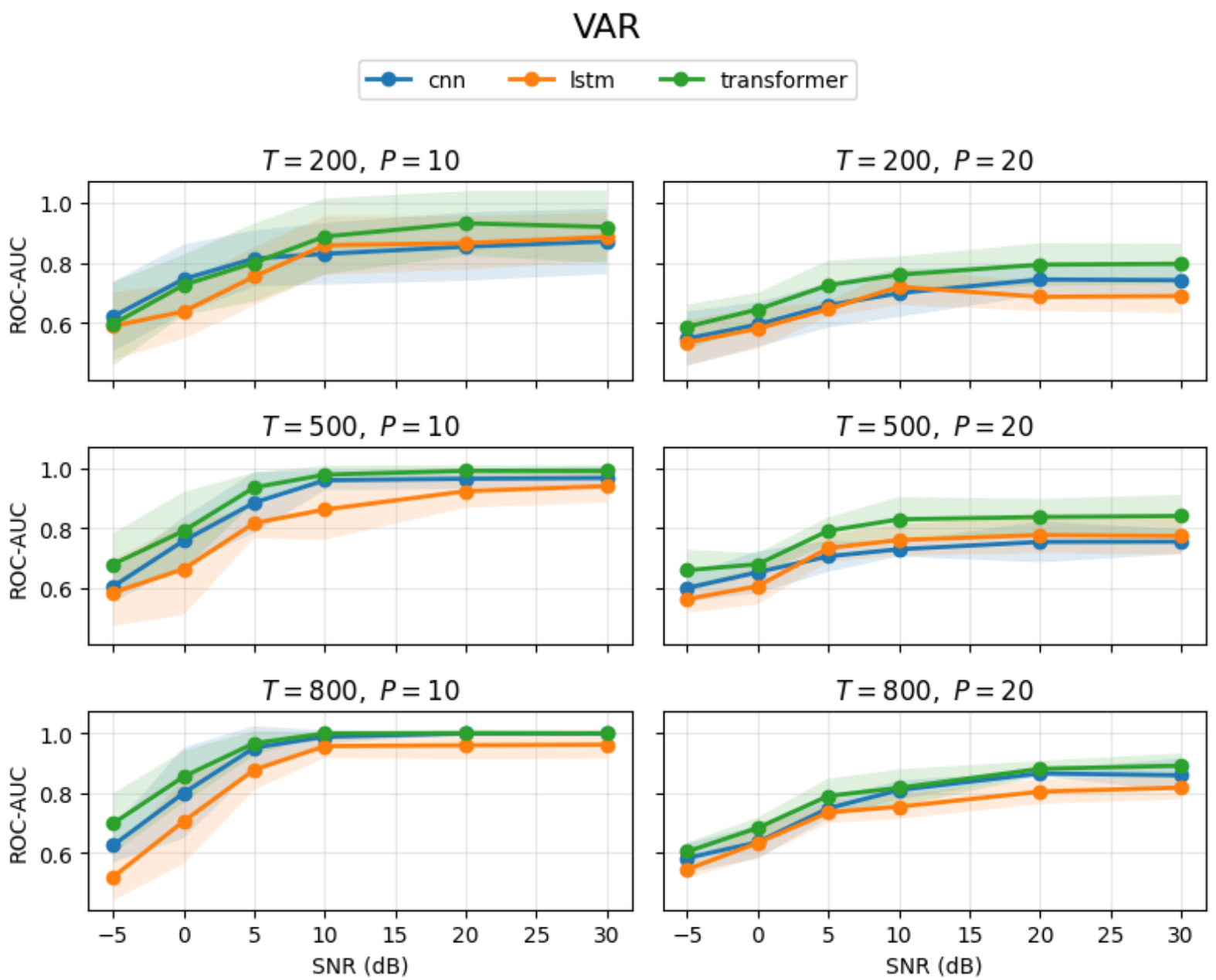
- Class imbalance grows harsher: identifying the true causal edges among a denser set of candidates is intrinsically harder.
- Denser graphs introduce more *spurious paths through mediating variables*, confusing the causal-discovery signal.

- **Transformer is the most resilient architecture** across density/SNR configurations, suggesting attention mechanisms may generalize better here than LSTM's recurrence or CNN's weight sharing (a direction worth separate investigation).

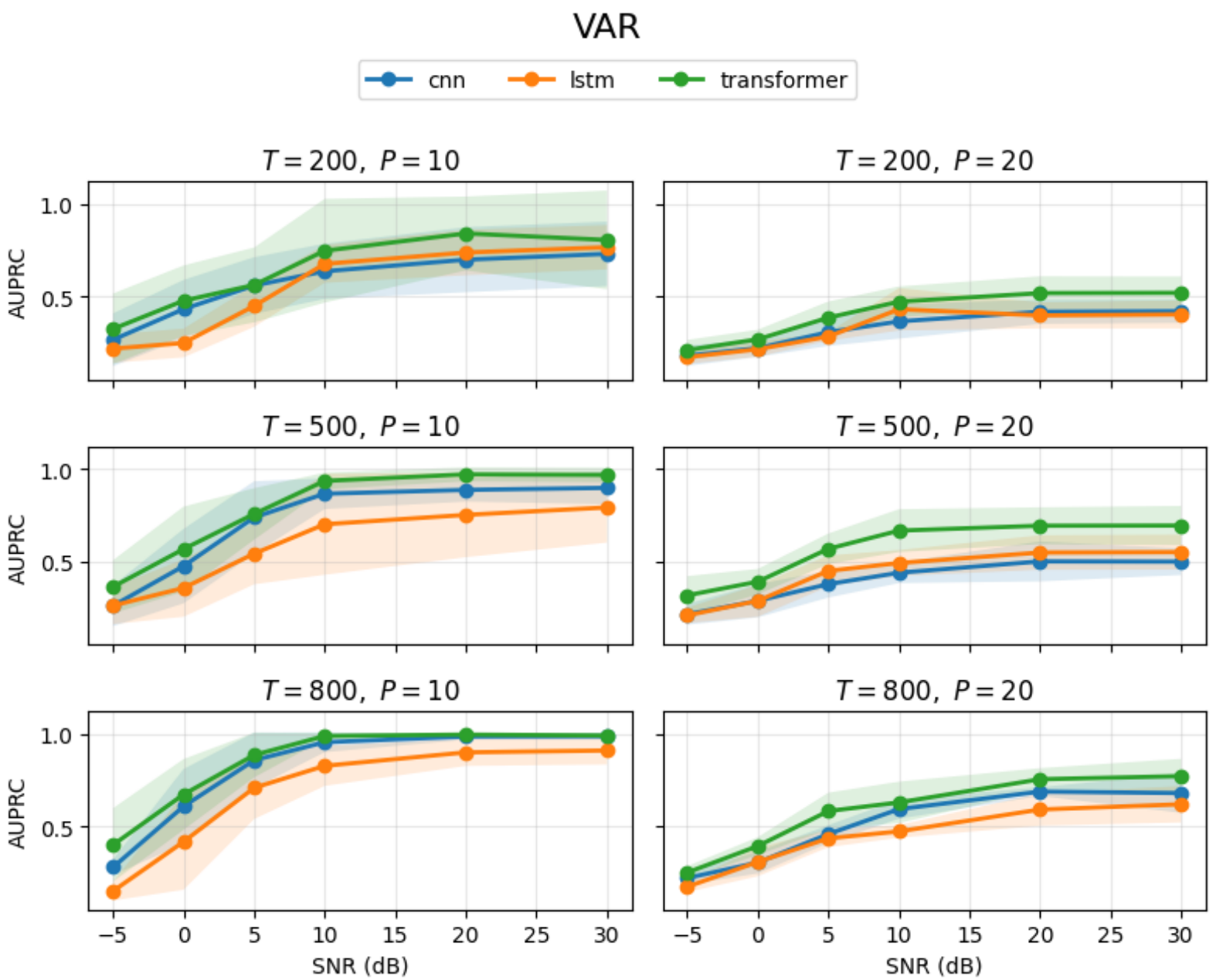
Takeaway

Denser causal structure and lower SNR both hurt precision-recall performance more than ROC performance; Transformer's robustness across these axes hints at attention as a key contributing factor.

VAR(2): Effect of T and P on AUROC



VAR(2): Effect of T and P on AUPRC



Sensitivity Analysis: Effect of P, T, and SNR

Setup: true Granger-causal edges confined to 20% of components (sparse ground truth); $P = \{10, 20\}$, $T = \{200, 500, 800\}$, SNR $= \{-5, 0, 5, 10, 20, 30\}$ dB, AR lag order $= 5$, 5 seeds, all models trained under **No-ILD**.

- **Transformer outperforms CNN and LSTM on average**, across essentially every (P, T, SNR) configuration tested.
- **Performance increases monotonically** with both SNR and time series length T, more signal and more data both make Granger-causal structure easier to recover.

- **Performance degrades substantially as P grows** from 10 to 20, a larger number of candidate series makes the discovery problem markedly harder for all architectures.
- **AUPRC is consistently lower than AUROC** (Figure 7b). Under the sparse ground-truth (class-imbalanced) setting, AUROC can *overstate* true performance, while AUPRC better exposes the real difficulty of correctly identifying the few true causal edges.

Takeaway

More data and higher SNR help recovery; more candidate series hurts it. AUPRC, not AUROC, is the more honest metric under sparse ground truth.

Model Architecture Sensitivity Analysis

- **Data:** synthetic 10-dimensional VAR(2) process with a known sparse Granger-causality graph used as ground truth.
- **Swept:** attention heads $h \in \{2, 4\}$, hidden width $n \in \{2, 8, 16, 32\}$, dropout $p \in \{0.2, 0.4, 0.6\}$ (capacity and regularisation raised together).
- **Measured:** train/val/test MSE, generalisation gap (val − train), and AUROC/AUPRC of the estimated GC matrix.
- **Protocol:** up to 200 epochs with early stopping; dashed line marks the selected epoch.

Training dynamics

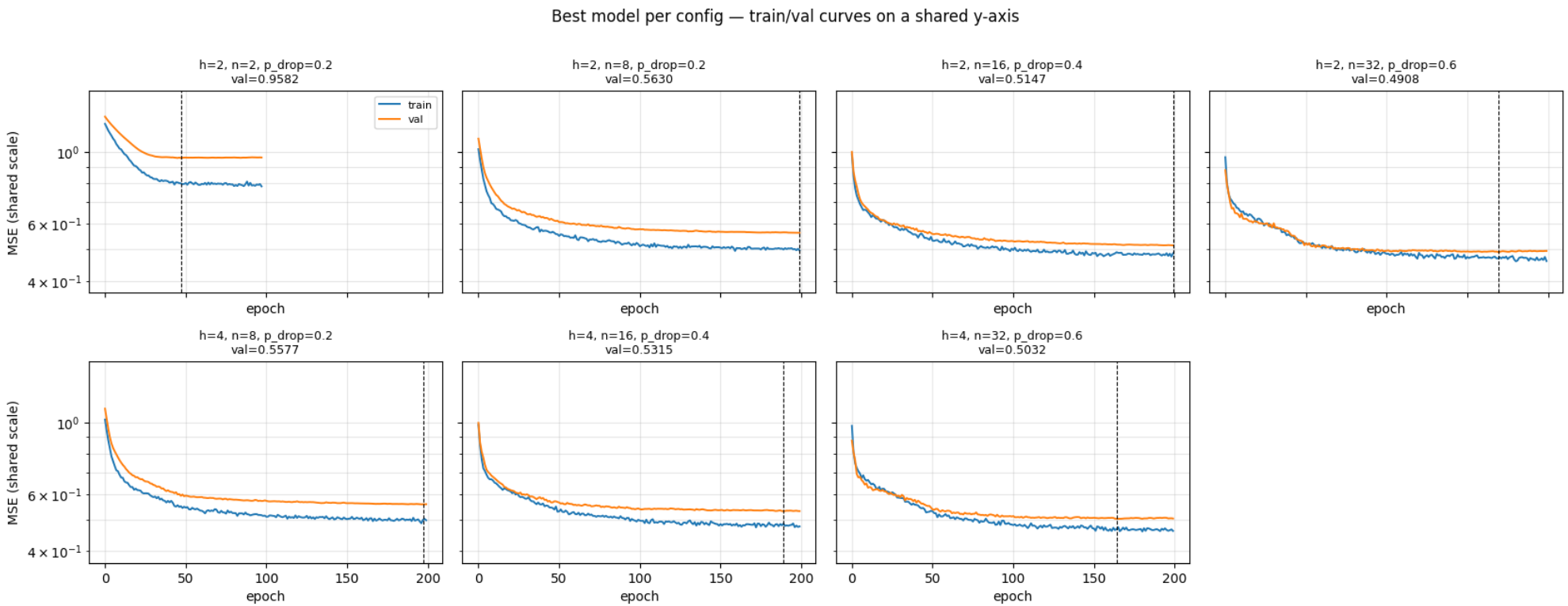


Figure: Train vs. validation MSE (log scale); dashed line = selected epoch.

- h=2, n=2 is capacity-starved: both curves flatten near MSE $\approx$ 0.8–0.95 by epoch 30, early stopping at epoch 48. It stops improving rather than overfitting.

- Every wider model is still descending at epoch 200; train and val curves stay parallel, so no configuration is in a runaway-overfitting regime.
- As dropout rises with width (n=16, p=0.4; n=32, p=0.6) the two curves converge , the widest models are the most tightly regularised.

Granger-causality recovery

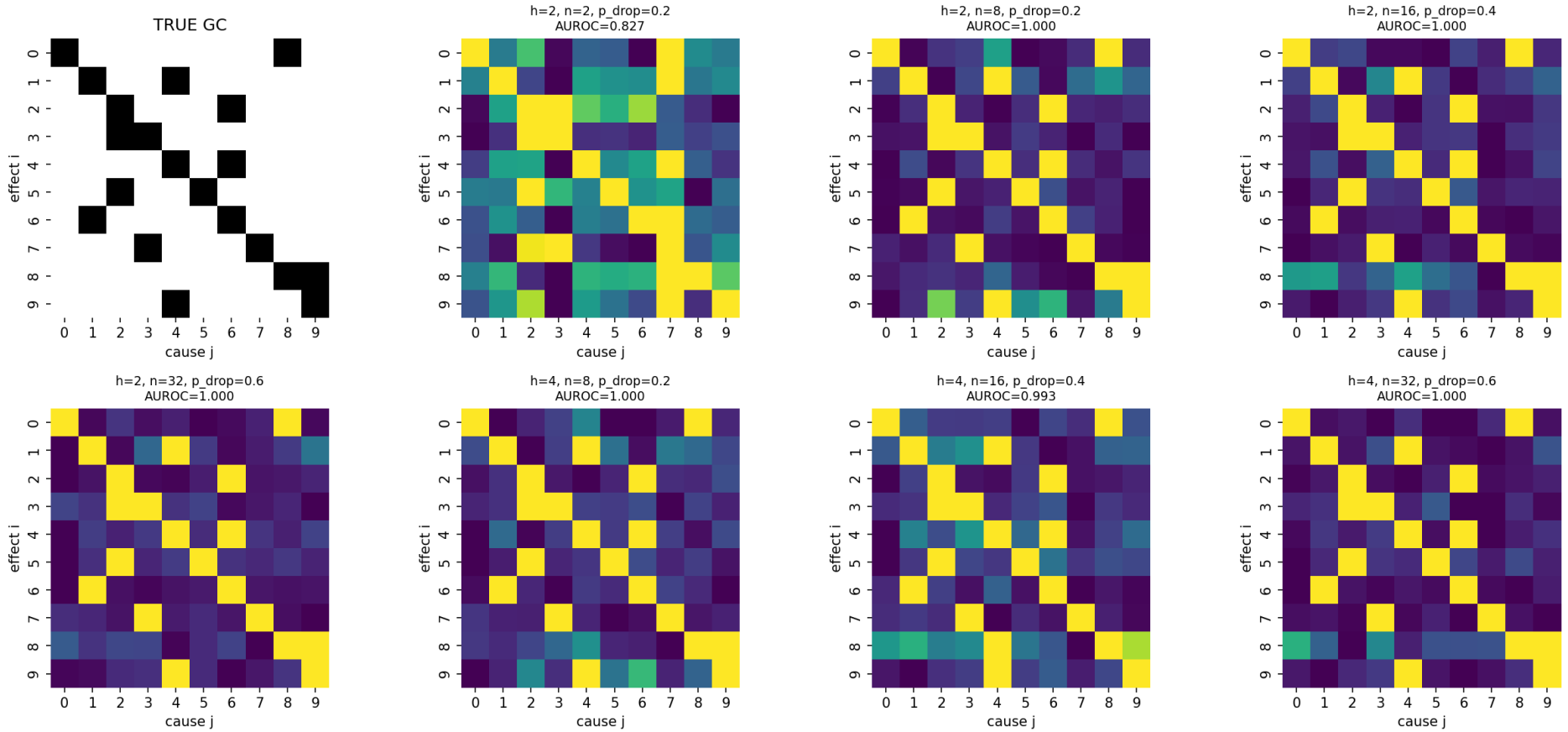


Figure: Estimated GC matrices vs. ground truth, with AUROC per configuration.

- Six of seven configurations reach AUROC $= 1.000$: the bright cells reproduce the true graph, including off-diagonal links ($0 \to 8$, $2 \to 3$, $4 \to 9$, $8 \to 9$).
- h=2, n=2 fails: AUROC 0.827, AUPRC 0.288; its map is diffuse, with mass smeared across columns 4–7 instead of on true parents.
- h=4, n=16 slips slightly (AUROC 0.993, AUPRC 0.901): row 8 picks up spurious mass on columns 0 and 7.
- Structure recovery saturates before forecast error does , capacity beyond n=8 buys little here.

Ranking and generalisation gap

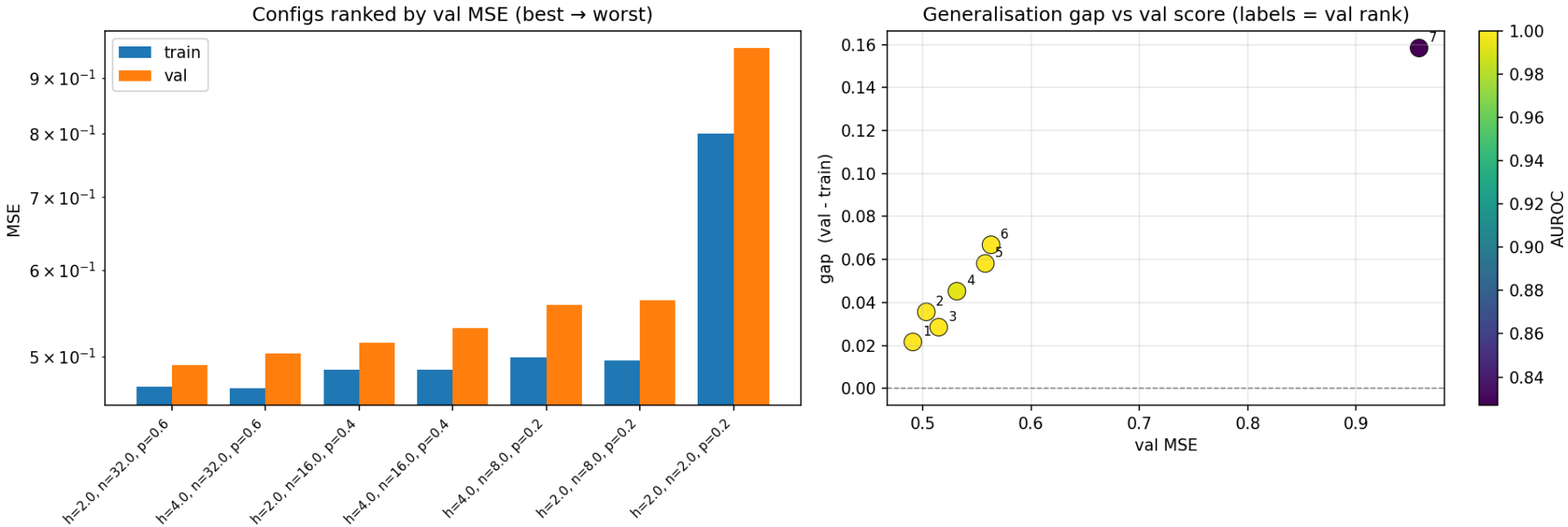


Figure: Configurations ranked by val MSE (left); gap vs. val MSE, coloured by AUROC (right).

- Ranking is monotone in width: $n{=}32 \succ n{=}16 \succ n{=}8 \succ n{=}2$; within each width $h{=}2$ edges out $h{=}4$. More heads do not help.
- Gap and val MSE move together (ranks 1–6 lie on a line): no accuracy-vs-overfitting trade-off to negotiate.
- Rank 7 ($h{=}2$, $n{=}2$) sits alone top-right , worst val MSE 0.958, largest gap 0.158 , and is the only point that fails causal discovery.

Full results

Rank	h	n	p	Train	Val	Gap	Test	Ep.	AUROC	AUPRC
1	2	32	0.6	0.469	0.491	0.022	0.520	170	1.000	1.000
2	4	32	0.6	0.468	0.503	0.036	0.539	165	1.000	1.000
3	2	16	0.4	0.486	0.515	0.028	0.567	200	1.000	1.000
4	4	16	0.4	0.486	0.531	0.045	0.574	190	0.993	0.901
5	4	8	0.2	0.500	0.558	0.058	0.578	198	1.000	1.000
6	2	8	0.2	0.496	0.563	0.067	0.577	200	1.000	1.000
7	2	2	0.2	0.800	0.958	0.158	0.987	48	0.827	0.288

Test MSE follows the same order as val MSE, so the ranking is not an artefact of validation-set selection (the ranks 5/6 inversion is within noise).

Takeaways

- **Use h=2, n=32, p=0.6:** best val MSE (0.491), smallest gap (4.6%), perfect AUROC/AUPRC.
- **Width matters, heads do not:** n=2 $\rightarrow$ n=32 roughly halves val error; h=2 $\rightarrow$ h=4 is neutral to slightly worse at every width.
- **Capacity, not overfitting, is the binding constraint:** only the smallest model has a large gap, and it underfits as well.
- **Caveats:** width and dropout were varied together, so their effects are confounded; most models were still improving at 200 epochs; single seed.

Results: Lorenz-96 (20-dim, Chaotic)

Model – Regime	**AUCROC**	**AUPRC**
LSTM – No ILD	1.00 ± 0.00	1.00 ± 0.00
CNN – No ILD	0.98 ± 0.01	0.91 ± 0.03
Transformer – No ILD	1.00 ± 0.00	1.00 ± 0.00
Transformer – ILD	1.00 ± 0.01	1.00 ± 0.01
GVAR	1.00 ± 0.00	1.00 ± 0.00
CUTS	0.96 ± 0.03	0.90 ± 0.02
cLSTM	0.93 ± 0.02	0.71 ± 0.04

- Even under chaotic dynamics ($F = 20$), unrestricted joint models excel
- CNN is the *most* affected by input dropout – likely due to **parameter sharing** across filters

Results: Netsim (fMRI BOLD, Data-Scarce)

Model – Regime	**AUCROC**	**AUPRC**
LSTM – No ILD	0.60 ± 0.07	0.17 ± 0.04
CNN – No ILD	0.60 ± 0.07	0.14 ± 0.04
Transformer – No ILD	0.60 ± 0.07	0.21 ± 0.08
GVAR	0.67 ± 0.06	0.26 ± 0.07
CUTS	0.72 ± 0.09	0.25 ± 0.08
cLSTM	0.62 ± 0.13	0.13 ± 0.04

- 15 dims, only $T = 200$ time points – **small-sample regime**
- Here sparse-regression methods (esp. CUTS) outperform simple joint models
- Still, joint prediction models provide a **reasonable baseline** even with scarce data

Case Study Setup: EEG Data

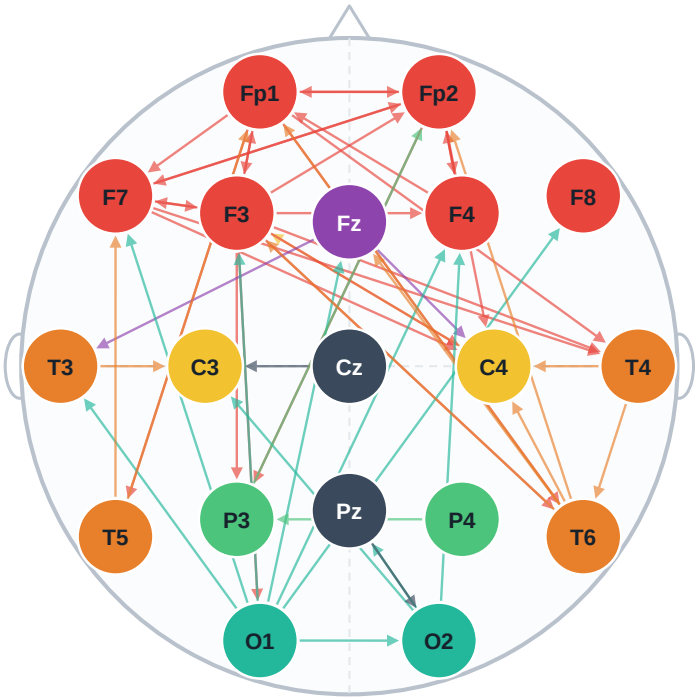
- Public dataset (Miltiadous et al., 2023): resting-state EEG, eyes closed
- 30 subjects: Alzheimer's disease vs. healthy controls
- 19 scalp channels, 3-minute window (1–4 min interval)
- **Model:** Transformer, 2 attention heads, 2 hidden layers $\times$ 128 neurons, dropout, trained **No-ILD** *separately per subject*
- Past **10 lags** used to predict 1-step-ahead for all 19 channels jointly
- Per-subject GC matrix extracted $\rightarrow$ averaged within each group $\rightarrow$ thresholded for binary connectivity

Reading the two networks

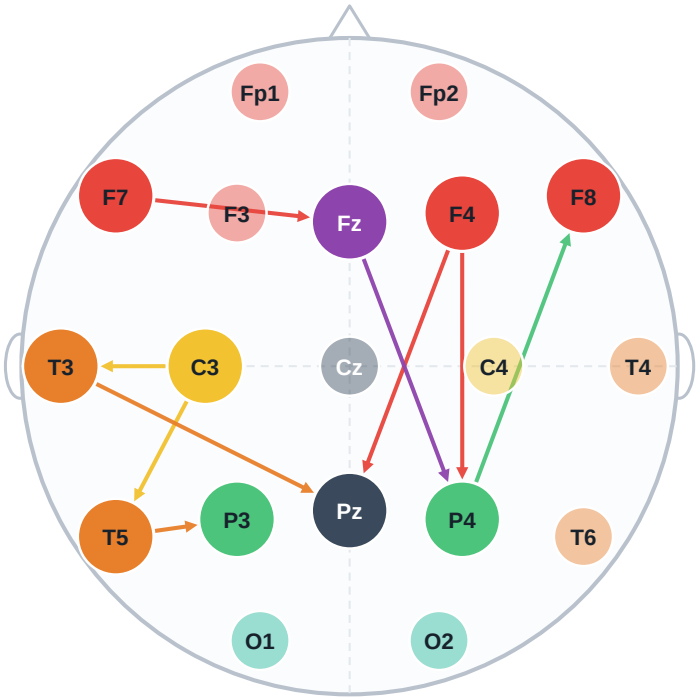
- **Healthy controls.** A dense, richly interconnected network with a hub at the frontal midline (Fz) and left-frontal channels (F7, F3, Fp1) driving nearly all other regions.
- **Alzheimer's.** Starkly sparser. Most long-range and frontal-hub connections are absent, leaving a handful of isolated edges.
- Region codes: LF/RF frontal, LT/RT temporal, LC/RC central, LP/RP parietal, LO/RO occipital, FM/CPM midline.

Directed edges unique to each group

Present in HC, absent in AD , 53 edges



Present in AD, absent in HC , 9 edges



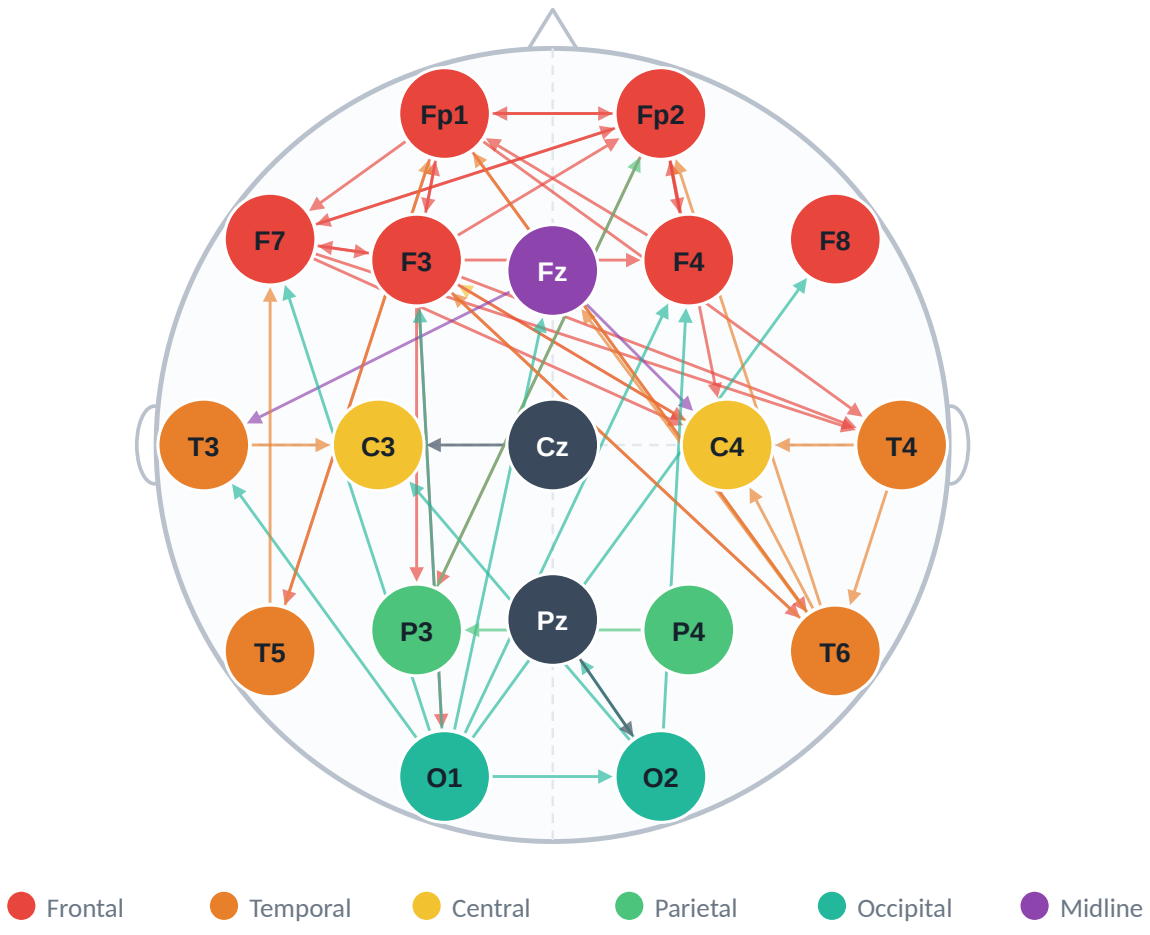


$A[i,j] = 1 \Rightarrow j \rightarrow i$; the 84 edges shared by both groups are not shown. Arrow colour denotes the source region.

Connections lost in Alzheimer's disease

53 of the 137 directed edges found in controls are absent in AD.

- **Frontal hub.** F3, F7, Fp1, Fp2 lose their mutual coupling; F3 alone loses 8 outgoing edges.
- **Frontal → right temporal.** Drive onto T4, T6 and C4 disappears.
- **Occipital feedback.** O1 and O2 no longer drive F3, F4, F7, F8, Fz or C3.

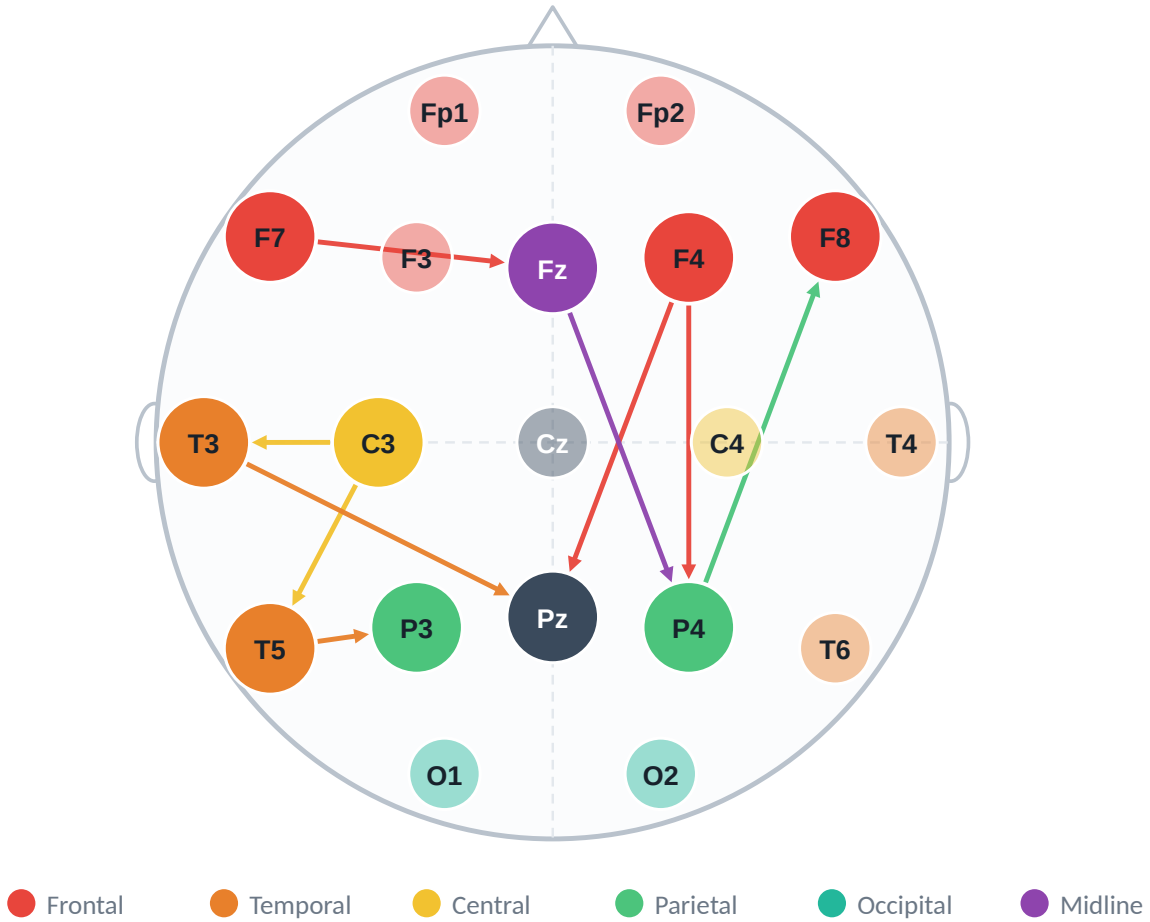


Connections present only in Alzheimer's disease

Only 9 of the 93 directed edges in AD are absent in controls.

- **Left centro-temporal.** C3→T3, C3→T5, T5→P3, T3→Pz.
- **Fronto-parietal.** F4→Pz, F4→P4, Fz→P4, P4→F8.
- **Anterior.** F7→Fz is the only frontal-to-frontal addition.

Short-range and left-lateralised: no gain of long-range coupling.



Summary

- HC: 137 directed edges AD: 93 shared: 84.
- **53 edges lost** versus **9 gained** , overwhelmingly a loss of connectivity, not a rewiring.
- Lost edges are long-range and anterior; gained edges are short-range and left centro-temporal.
- Occipital $\rightarrow$ frontal feedback, intact in controls, is absent in AD.

Sensitivity Analysis: EEG Granger Causality

How much of the Alzheimer's–Healthy connectivity gap survives scrutiny?

A stress test of the discovered Granger-causal EEG connectivity difference against three modeling choices: network architecture, training budget, and detection threshold.

- 19 EEG channels
- 14 Alzheimer's / 15 Healthy subjects
- Prediction horizons: lag 10, lag 20
- 9 architectures: $h \in \{32, 64, 128\} \times l \in \{1, 2, 3\}$

Method

One pipeline, swept 9 ways per subject

Each subject's EEG is fit by a Transformer predicting each channel from the other 18. Masking one input channel across 200 Monte-Carlo dropout passes and comparing prediction error gives a probability that channel j Granger-causes channel i: $\mathrm{GC}[i, j] = P(j \to i)$.



- hidden $\in \{32, 64, 128\}$; layers $\in \{1, 2, 3\}$
- dropout $\in \{0.2, 0.3, 0.5, 0.7\}$, best kept per structure
- lag $\in \{10, 20\}$
- 50 epochs, Adam, lr $1\mathrm{e}{-2} \to 1\mathrm{e}{-3}$ at epoch 25

Validation check (Study 01)

50 epochs is enough for small models, not for h=128

Slope of validation MSE over the last 10 epochs: near zero means converged; clearly negative means still improving when training was cut off.

- h32, l1–3: best epoch 44–45/50, slope -0.00001 to -0.00008, converged
- h64, l1–2: best epoch 41–44/50, slope -0.00012 to -0.00015, converged
- h64, l3: best epoch 14–15/50, slope $+0.0002$ to $+0.0009$, diverging
- h128, l1: best epoch 32–34/50, slope -0.00015 to -0.00040, still improving
- h128, l2–3: best epoch 22–28/50, slope -0.0002 to -0.0012, still improving

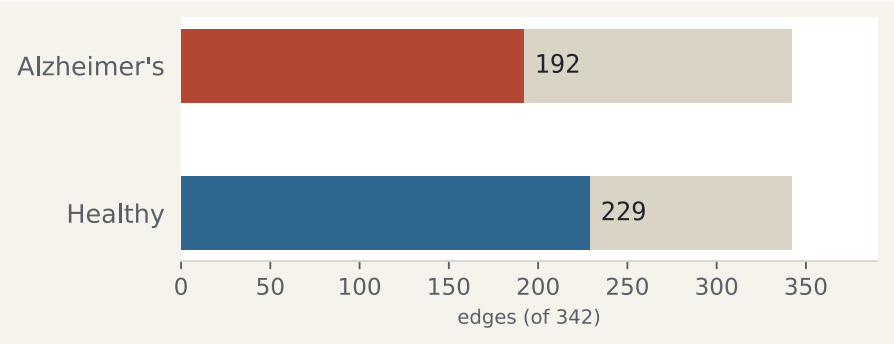
Two distinct failure modes, not one: **h128** is genuinely under-trained, its poor MSE (Study 02) is partly a budget artifact, not purely a complexity effect. **h64,l3** instead diverges early after a good epoch 14, an optimization instability, most likely the initial lr=1e-2 being too aggressive for that shape.

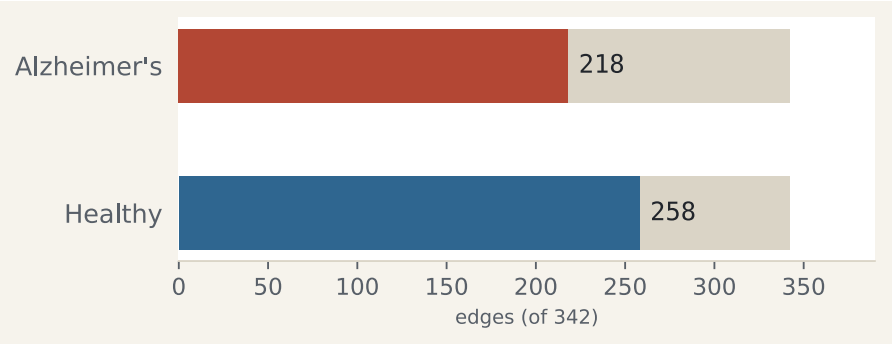
Finding (Study 02)

Using one validated architecture per lag, Healthy connectivity is consistently denser

Mean GC matrix computed across subjects first, then thresholded at the τ that best separates the two group means, no per-subject or per-architecture cherry-picking.

- Lag 10: $h = 32$, $l = 1$, $\tau^* = 0.99$, $\Delta = -37$ edges (of 342)
- Lag 20: $h = 64$, $l = 1$, $\tau^* = 0.99$, $\Delta = -40$ edges (of 342)

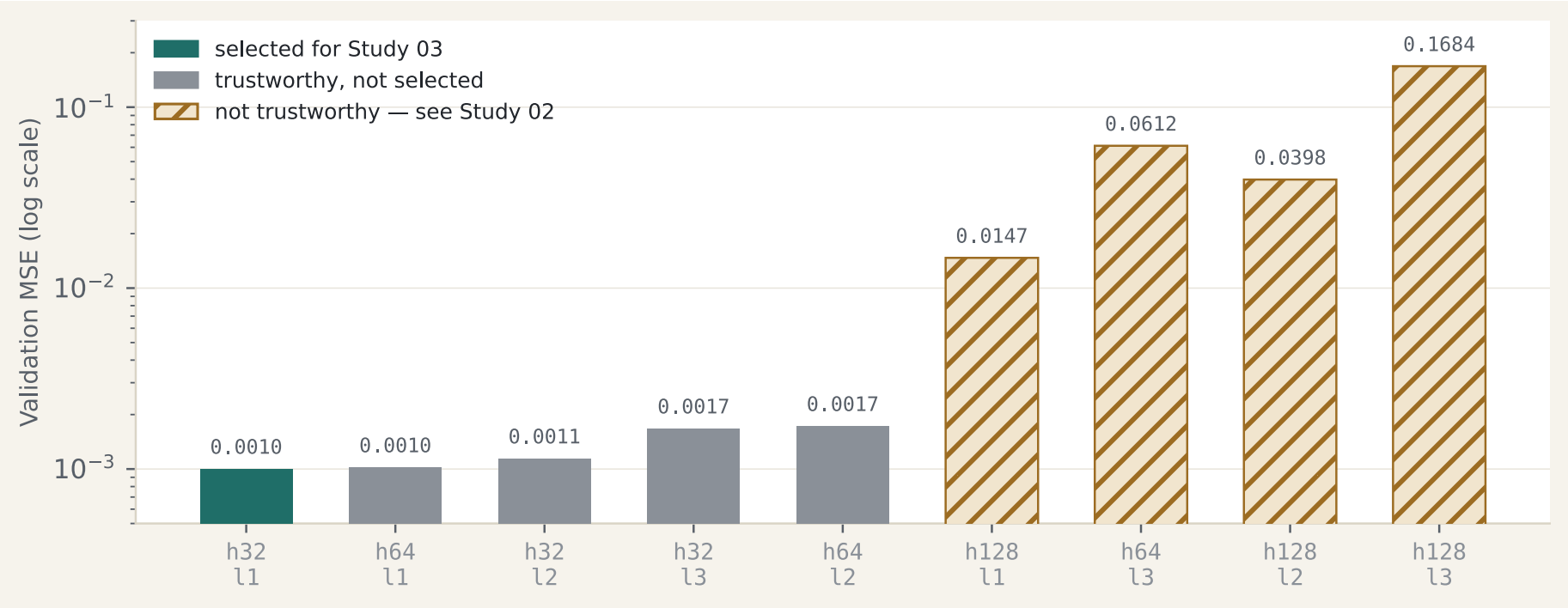




Justification (Study 02 continued)

Simple architectures aren't a compromise,they fit best

Mean validation MSE, lag 10, sorted by model complexity (hidden $\times$ layers), bar height log-scaled. Error rises sharply once complexity exceeds $h = 64$ at 2 layers.

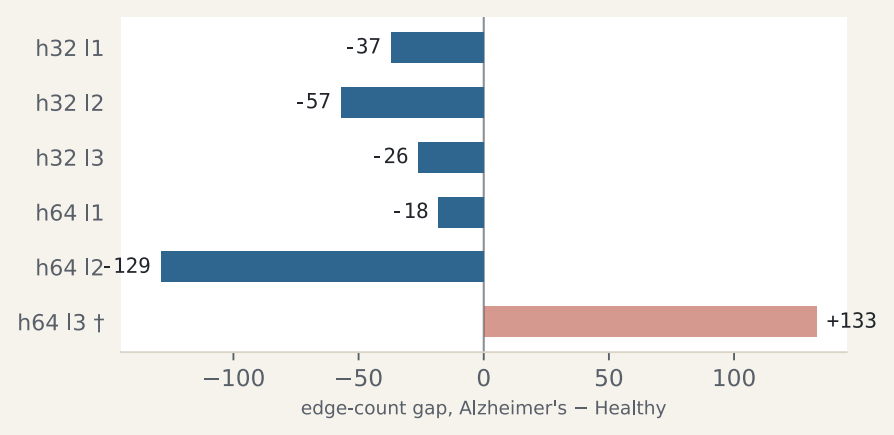


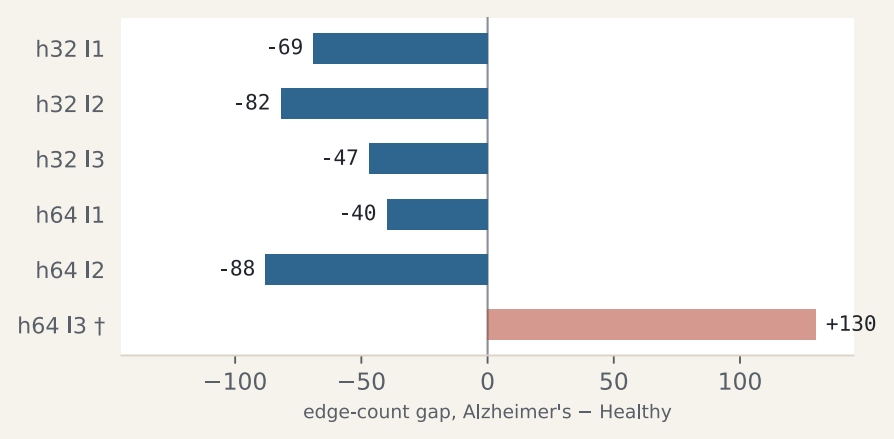
- Bar height is log-scaled (raw MSE spans 0.0010–0.1684) so the low end stays visible.

Robustness (converged architectures only)

Direction holds in all 10 converged cells,only the diverging architecture disagrees

Edge-count gap (Alzheimer's − Healthy) at each architecture's own best-discriminating τ, group-first mean GC. 10 of 12 tested (architecture, lag) cells agree; the 2 that don't are both h64,l3,the architecture Study 02 flagged as diverging.





- Left: lag 10. Right: lag 20.
- h64,l3 excluded from the "converged" claim.
- Blue extends right = Healthy > AD; red extends left = AD > Healthy; shared scale, ±140 edges.

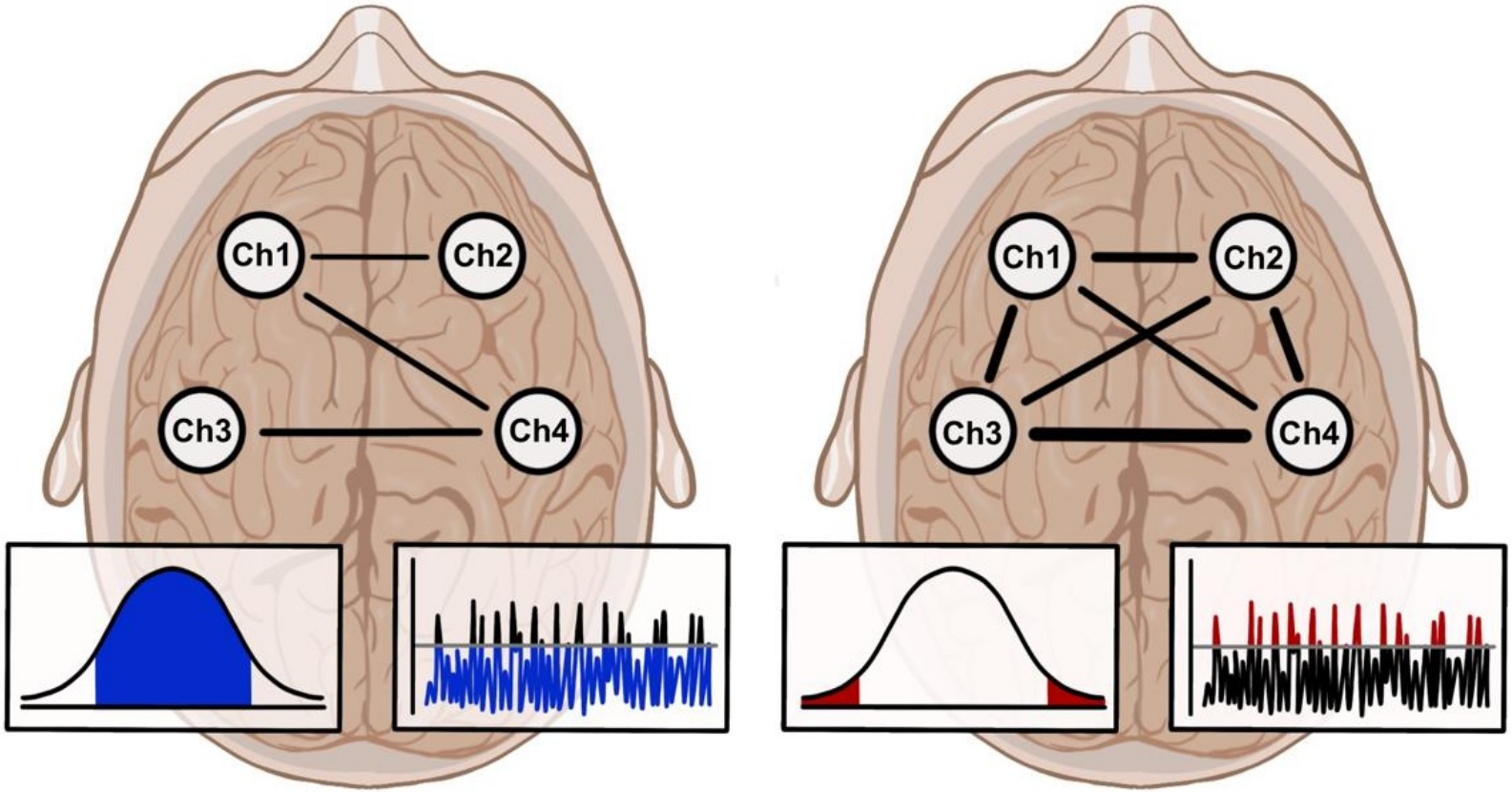
Outline

1. Deep learning / Granger Causality
   with Malik Shahid Sultan
2. Extremal Vulnerability
   with Paolo Redondo
3. Topological Effective Connectivity Modeling
   with Anass El-Yaagoubi

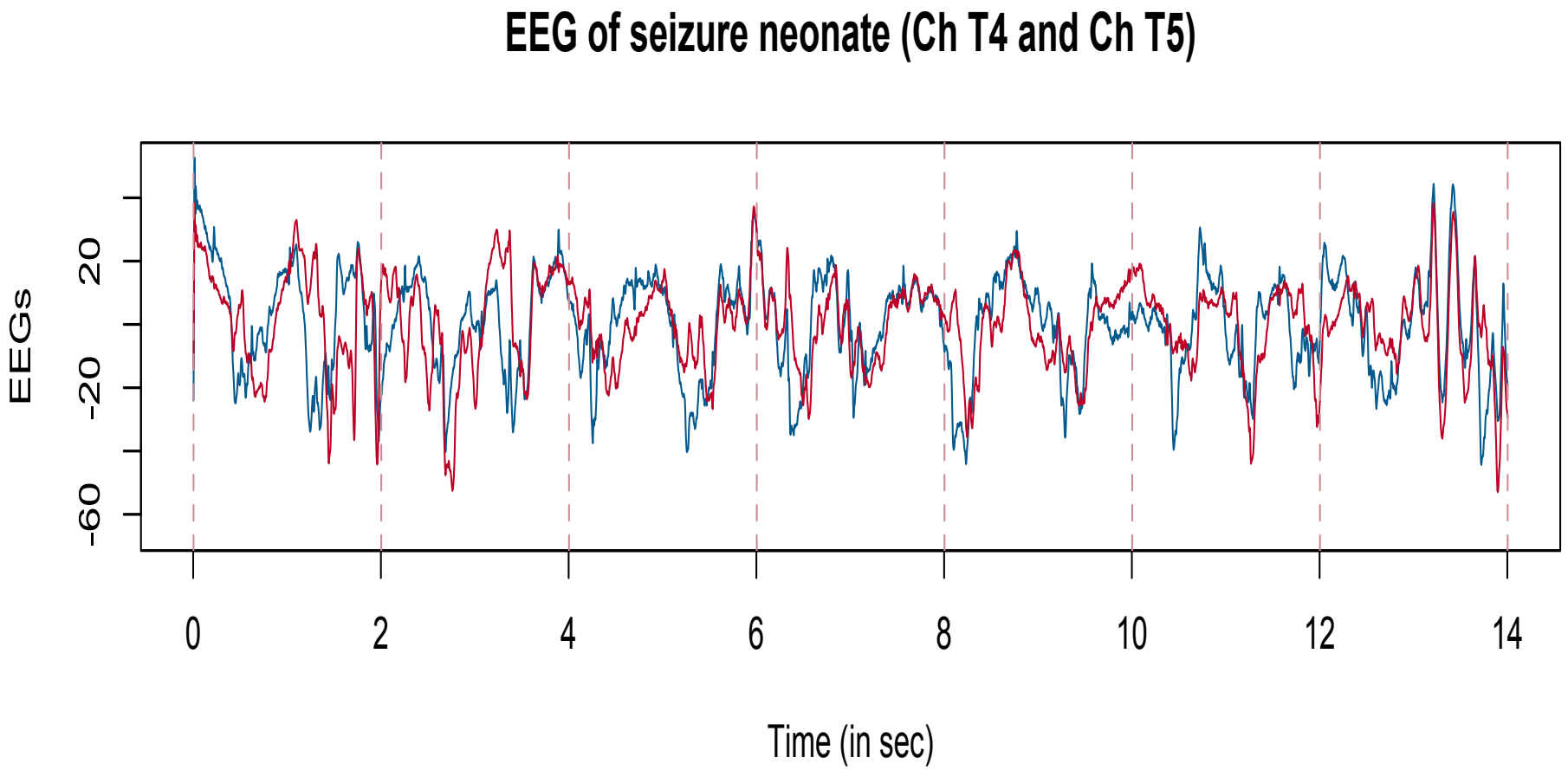
Extremal Dependence

- Dependence in the "bulk" vs at the "tails"
- Financial market - stock prices during bull vs bear vs market crash
- When the ship sinks - everyone is thrown into the ocean!
- Motivation: identify specific sectors that are systematcially exposed to severe market downturns

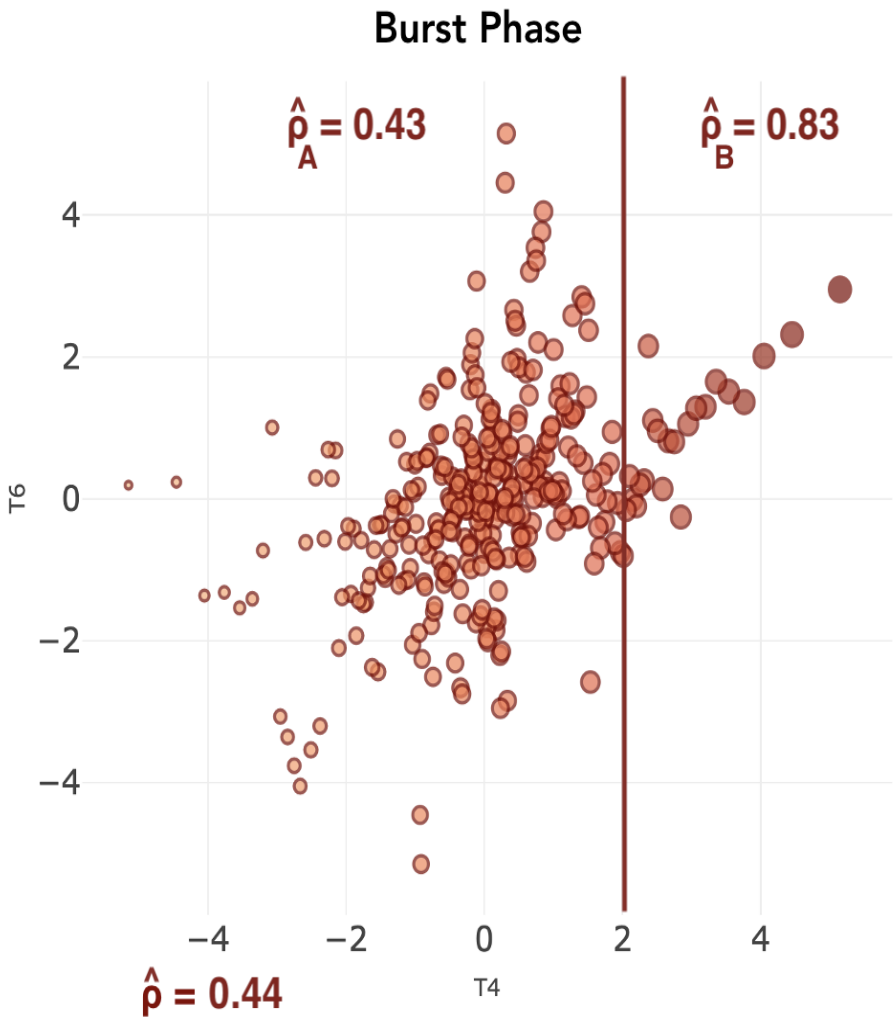
Extremal Dependence

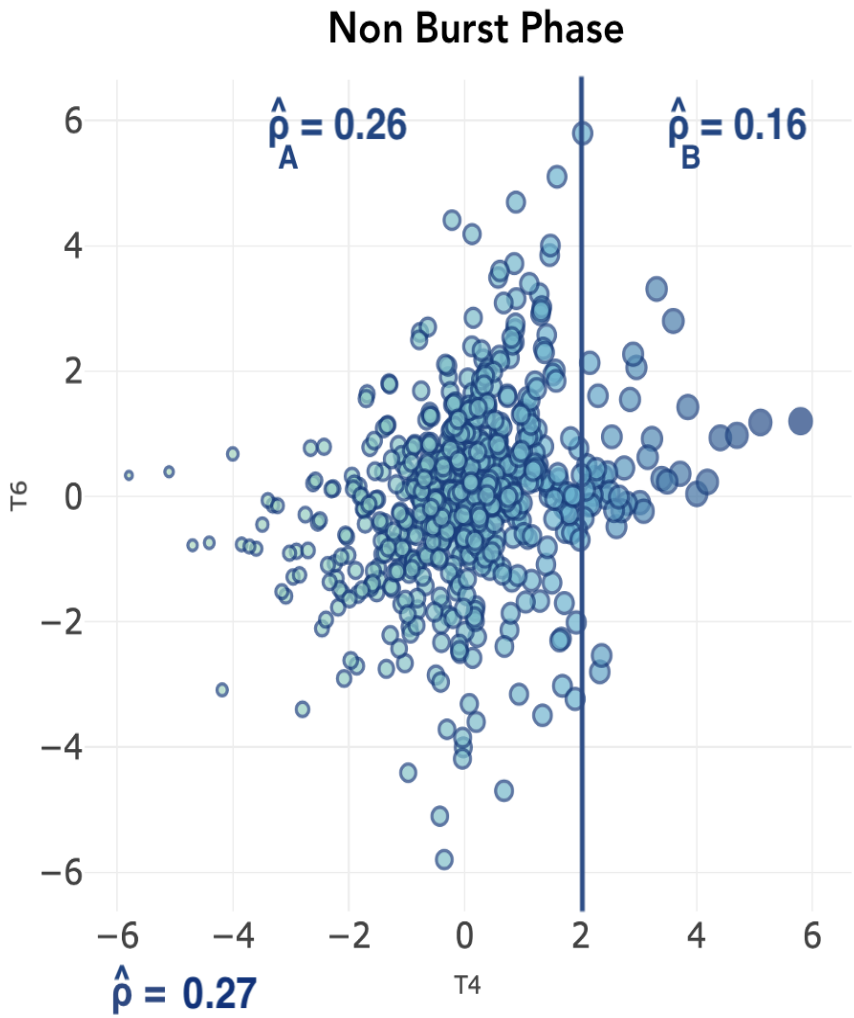


Extremal Dependence - Neonate EEG

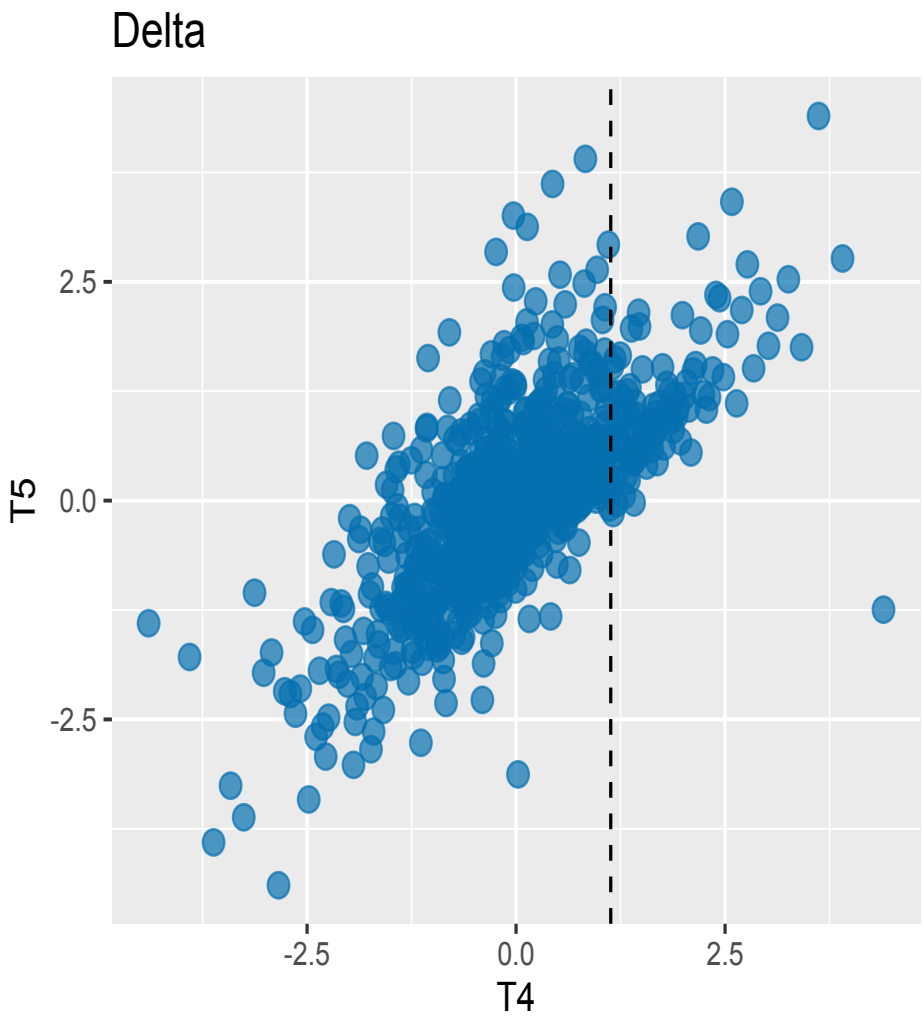


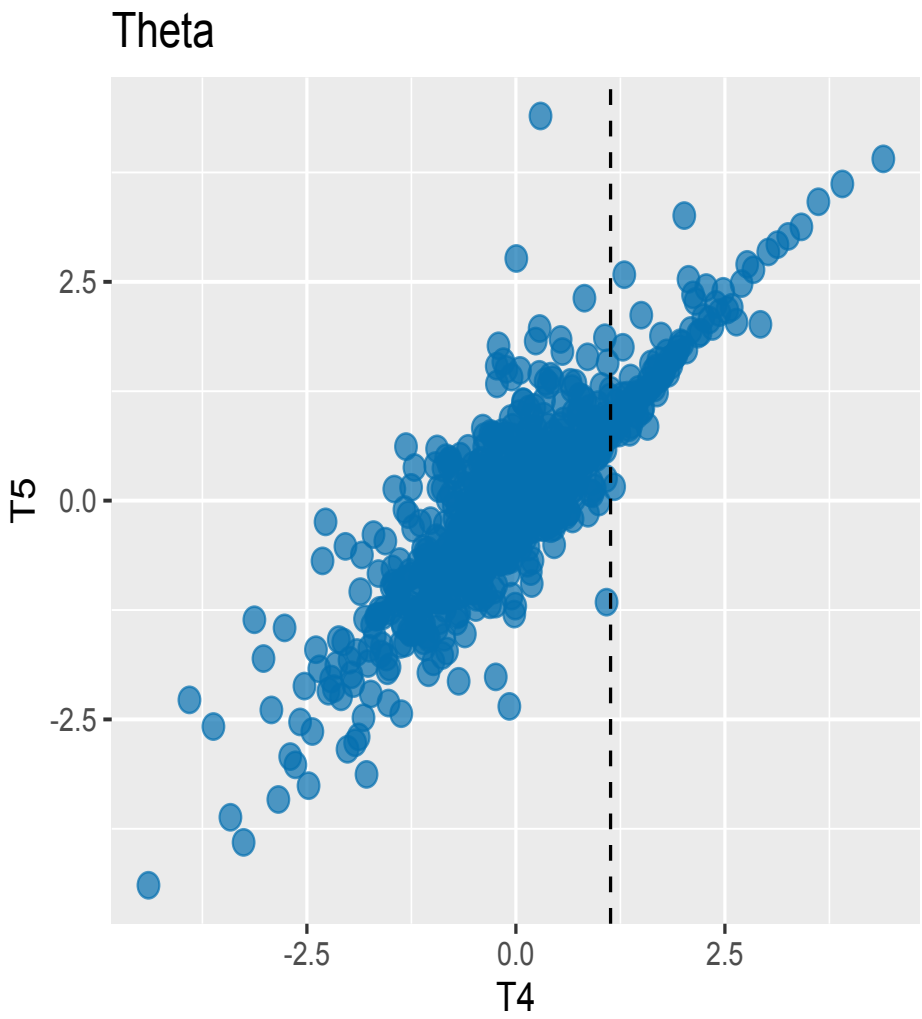
Extremal Dependence - Neonate EEG



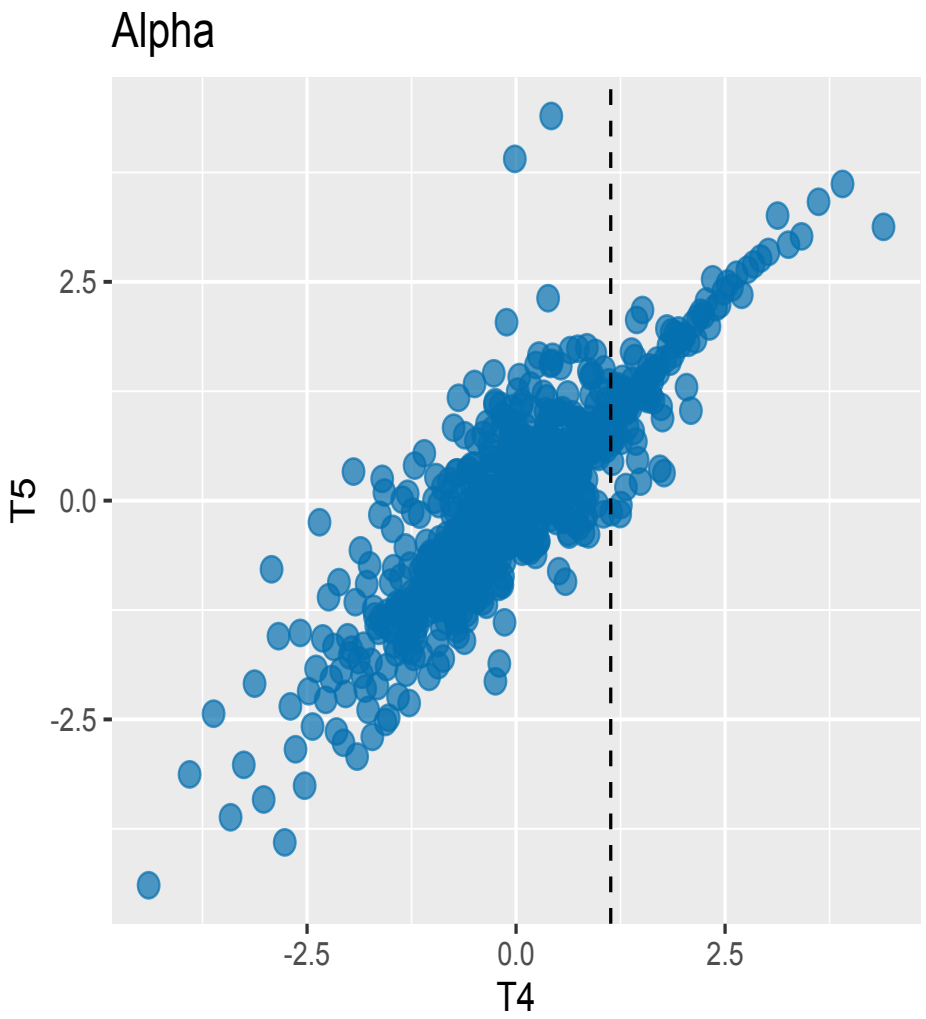


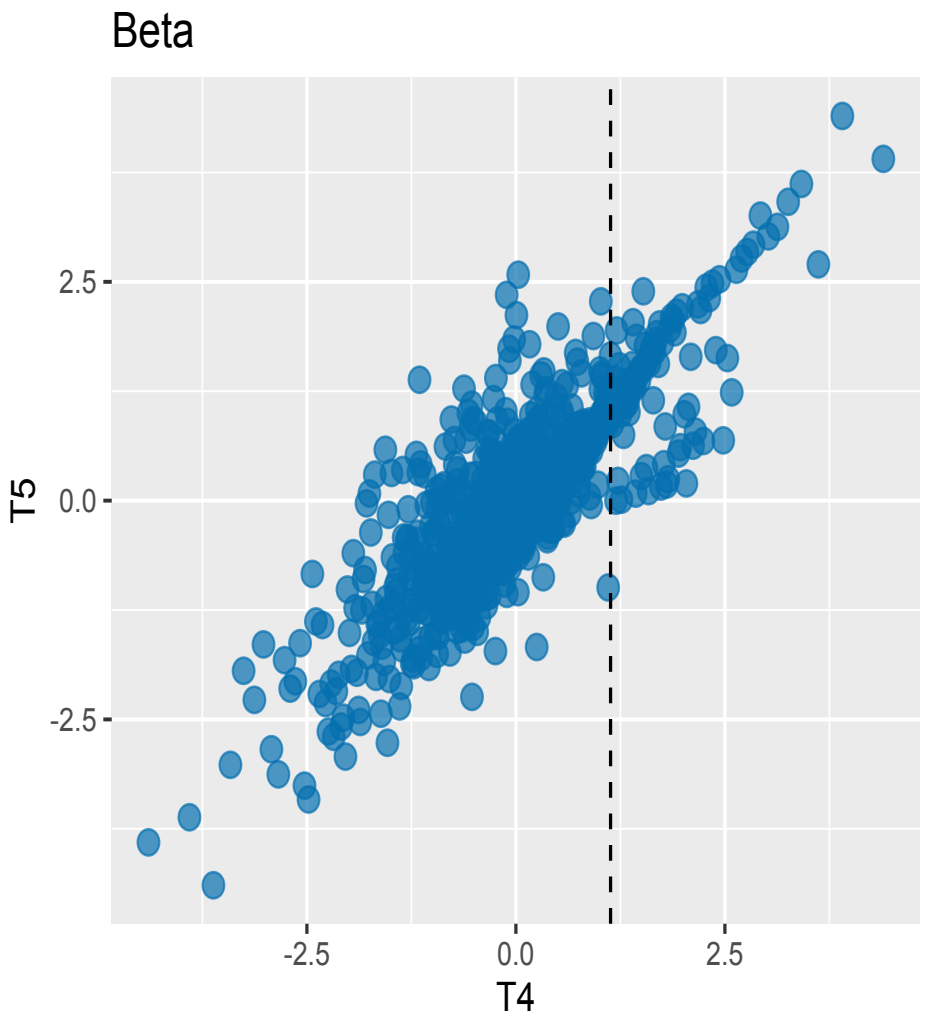
Extremal Dependence - Neonate EEG



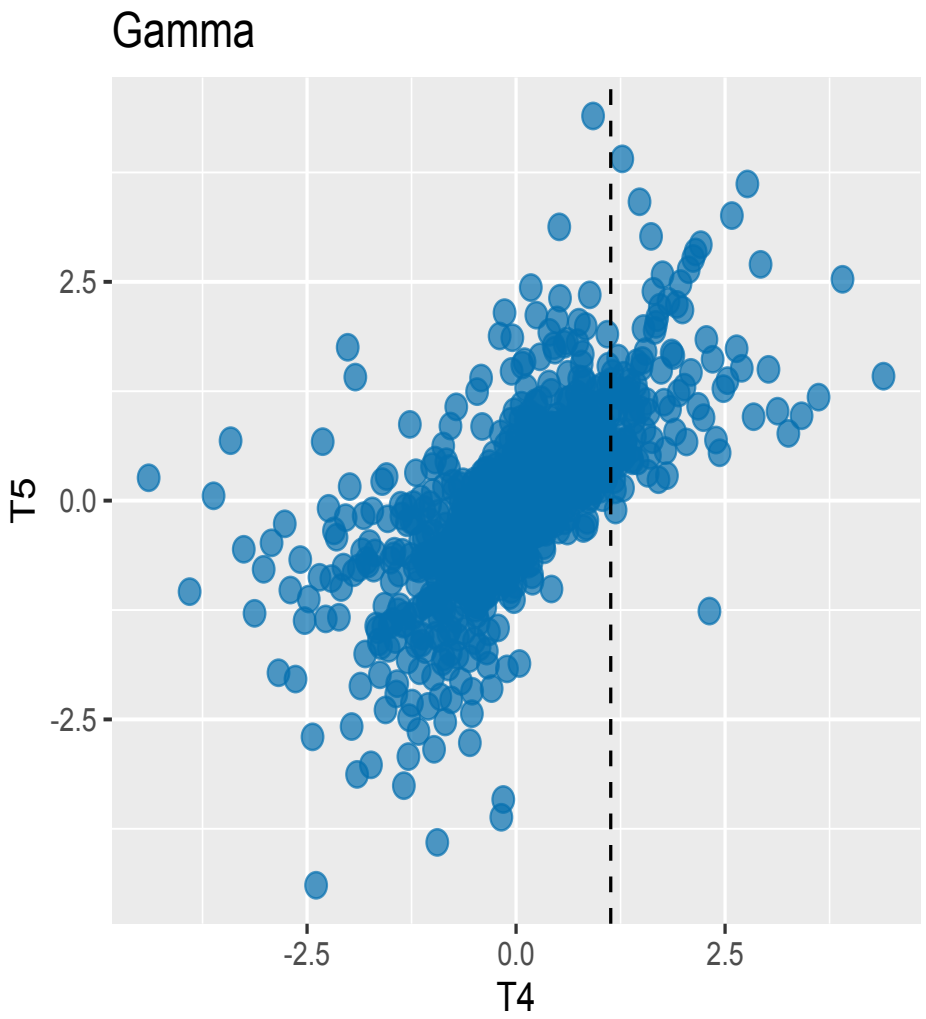


Extremal Dependence - Neonate EEG





Extremal Dependence - Neonate EEG



Talento et al. papers

- Spectral extremal connectivity of two-state seizure brain waves. Under revision.
- Canonical tail dependence for soft extremal clustering of multichannel brain signals. Under revision.
- KenCoh: A ranked-based canonical coherence. Under revision.

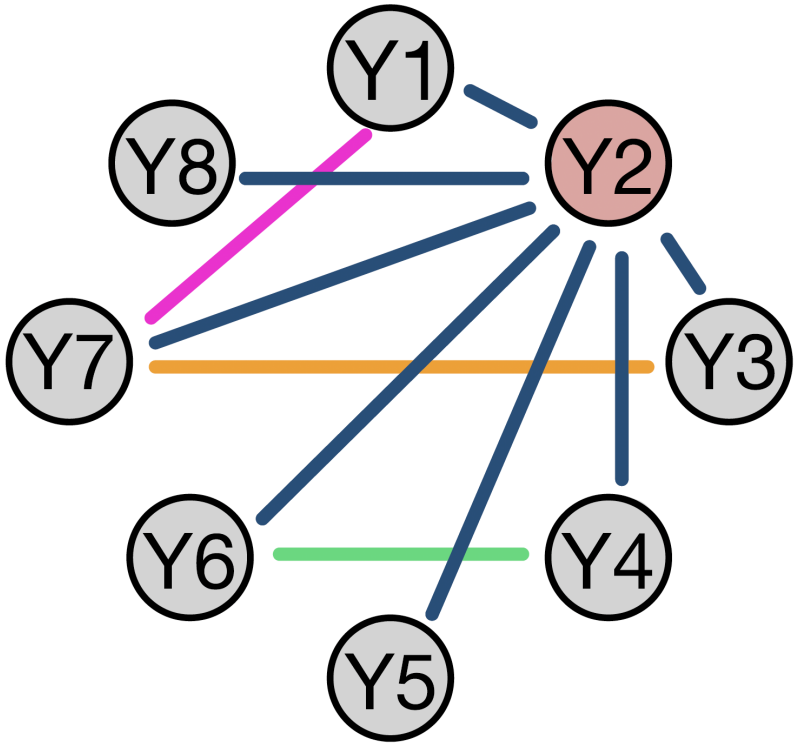
Extremal Vulnerability: Motivation

- During market stress: extreme losses occur simultaneously across assets
- Need to understand systemic risk: identify assets that are more vulnerable
- Vulnerable assets: those that experience repeated extreme loss
- Extremal vulnerability - model long run propensity of a component to experience extremes, when other components do.
- Novelty: formal EVT framework for assessing a component's susceptibility in a closed system

Extremal Vulnerability[1]

Consider a system consisting of d components generating $\boldsymbol{Y} = (Y_1, \ldots, Y_d)^\top$, where $Y_i = \{Y_{i,t} \, ; t \in \mathbb{Z}^+\}, i = 1, \ldots, d$.

Example: Suppose the tail dependencies between the components of $\boldsymbol{Y}$ are summarized as below.



- In principle, this makes Y_2 more vulnerable to extreme events than the rest.

[1] Redondo, P. V. *et al.* (2026+). Extremal Vulnerability. In preparation.

Extremal Vulnerability (Motivation)

Goal: Develop a principled framework to rank components w.r.t. extremal vulnerability.

Starting point: the Tail Dependence Matrix (TDM), defined as

$$\boldsymbol{A} = \{a_{i,j}\}, \qquad a_{i,j} = \lim_{u \to 1} \mathrm{P}\{Y_i > F_i^{-1}(u) \mid Y_j > F_j^{-1}(u)\}.$$

Here, F_i^{-1} and F_j^{-1} are the quantile functions of Y_i and Y_j, respectively.

- $a_{i,j} > 0 \implies Y_i$ and Y_j are asymptotically dependent
- $a_{i,j} = 0 \implies Y_i$ and Y_j are asymptotically independent

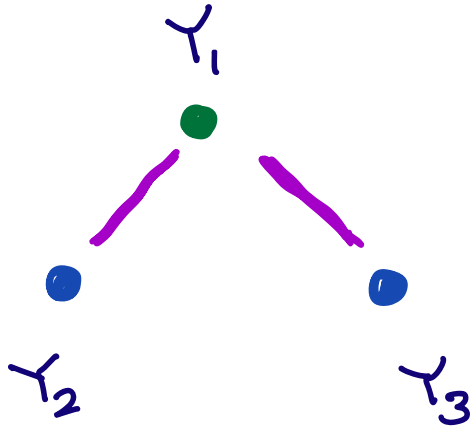
Proposal: Extremal Vulnerability Rank (XVR) Algorithm

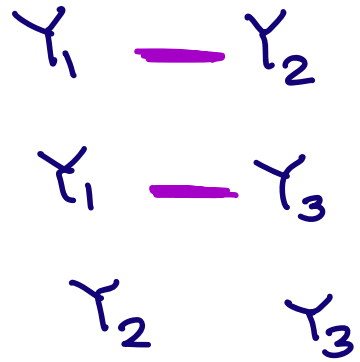
- Develop a single summary measure that captures the relative importance of each component within a system's extremal dependence structure.
- XVRank is inspired by the PageRank algorithm[2], but replacing hyperlink structure with extremal dependence measures (via TDMs).
- Output: identifies the component, most susceptible to extreme events, accounting for interactions with all other components.
- Interpretation of scores: long-run propensity of a component to be affected by extremes occurring within the system.

[2] Gleich, D. F. (2015). PageRank beyond the web. *siam REVIEW* **57**, 321–363.

Extremal Vulnerability Rank (XVR): Core Ideas

Co-existence of Extreme Events





Y_1 is "more vulnerable" than either

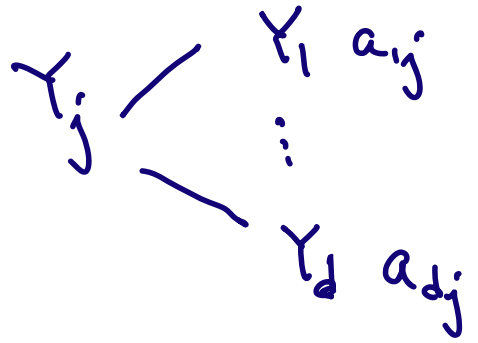
Y_2 or Y_3

Extremal Vulnerability Rank (XVR): Core Ideas

- Recall: (a_{ij}) A – tail dependence matrix TDM

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1d} \\ \vdots & & a_{ij} & & \vdots \\ a_{d1} & & a_{dj} & & a_{dd} \end{bmatrix}$$

- Define $P_{ij} = \dfrac{a_{ij}}{\sum_{l=1}^{d} a_{il}}$



- $P = \{P_{ij}\}$
- $P = A\left[\mathrm{diag}(\mathbb{1}^T A)\right]^{-1}$

Extremal Vulnerability Rank (XVR): Core Ideas

PROPOSITION

- Let $S = \{ Q \in \mathbb{R}_+^{d \times d} : \mathbb{1}^T Q = \mathbb{1}^T, \; Q \geq 0 \}$
- Remark: $P \in S$
- Under the Bregman Divergence (KL),

$$\text{BregProj}(A) = P$$

Extremal Vulnerability Rank (XVR): Core Ideas

- Consider a MC $\{Z_s\}$ ~ pairwise co-movement of extremes in $\{Y\}$
- Initial distn $\pi_0 = [\pi_{01}, \ldots, \pi_{0d}]$

  π_{0i} = Prob extreme event initially at i
- Assume P irreducible & aperiodic
- Then $\pi_h = P\, P^{h-1}\, \pi_0$

Extremal Vulnerability Rank (XVR): Core Ideas

- $\lim_{h \to \infty} \pi_h := \pi = (\pi_1, \ldots, \pi_d)$

π_i = long-run probability of an extreme occurence in the i^{th} component, given that extreme events arise across components of Y

Extremal Vulnerability Rank (XVR): Core Ideas

DEFINITION 1 of Extremal Vulnerability

- $A = (a_{ij})$ Tail Dependence Matrix
- $P = (p_{ij})$ $A\left[\mathrm{Diag}(\mathbf{1}^T A)\right]^{-1}$
- P irreducible & aperiodic
- Define $\pi = (\pi_1, \ldots, \pi_d)$ to be the unique vector satisfying $P\pi = \pi$

$\Rightarrow$ π extreme vulnerability vector

Extremal Vulnerability Rank (XVR): Core Ideas

DEFINITION 2 Generalized Extreme Vulnerability

- Recall $a_{ij} = \lim_{u \to 1} P(Y_i > F_i^{-1}(u) \mid Y_j > F_j^{-1}(u))$

- Define $a_{ij}(u) = P(Y_i > F_i^{-1}(u) \mid Y_j > F_j^{-1}(u))$

- Define $P(u) = A(u)\left[\text{Diag}\left(\mathbb{1}^T A(u)\right)\right]^{-1}$

$$\ldots \quad P(u)\,\pi(u) = \pi(u)$$

$$\pi = \lim_{u \to 1^-} \pi(u)$$

Gen Ext Vulnerability

Extremal Vulnerability (Inference)

The XVRank Algorithm:

1. For each $i = 1, \ldots, d$, let $R_{i,t}$ be the rank of $Y_{i,t}$ among $\{Y_{i,1}, \ldots, Y_{i,n}\}$. Then, estimate the TDM $\boldsymbol{A}$ using the data analogue

$$\widehat{\boldsymbol{A}} = \{\widehat{a}_{i,j}\} \quad \text{with} \quad \widehat{a}_{i,j} = 2 - \widehat{\ell}_{i,j}(1,1),$$

where the empirical stable tail dependence function $\widehat{\ell}_{i,j}$ is defined as

$$\widehat{\ell}_{i,j}(x,y) = \frac{1}{k} \sum_{t=1}^{n} I(R_{i,t} > n - kx \text{ or } R_{j,t} > n - ky).$$

2. Calculate the Bregman projection of $\widehat{\boldsymbol{A}}$ onto $\mathcal{S}$, denoted by $\widehat{\boldsymbol{P}}$, as $\widehat{\boldsymbol{P}} = \widehat{\boldsymbol{A}} \operatorname{diag}(\mathbf{1}^\top \widehat{\boldsymbol{A}})^{-1}$.
3. Obtain the eigenvector of $\widehat{\boldsymbol{P}}$ that corresponds to the eigenvalue one and scale it so that its elements sum up to one. Denote the resulting eigenvector by $\widehat{\boldsymbol{\pi}} = (\widehat{\pi}_1, \ldots, \widehat{\pi}_d)^\top$, which we now call the XVRank scores.

Extremal Vulnerability (Inference)

The XVRank scores offer several advantages:

- They provide a principled framework to summarize extremal dependence by aggregating the information contained in the TDM into a single interpretable ranking vector.
- Because the scores arise as a stationary distribution, they admit a clear probabilistic interpretation.
- That is, $\widehat{\pi}_i$ represents the long-run probability that the i-th component is extreme when extremes propagate through the network according to the transition dynamics governed by $\widehat{\boldsymbol{P}}$.
- They obey appealing theoretical properties which we discuss below.

Extremal Vulnerability (Theoretical Guarantees)

- Assume that $k \equiv k_n$ is an intermediate sequence, i.e., $k = k_n \to \infty$ and $k_n/n \to 0$ as $n \to \infty$.
- Let $F \in D^d(G)$ be the joint distribution function F lies in the domain of attraction of a d-variate extreme value distribution G, i.e., there exist normalizing sequences $\boldsymbol{a}_n \in (0, \infty)^d$ and $\boldsymbol{b}_n \in \mathbb{R}^d$ such that

$$\lim_{n\to\infty} F^n(\boldsymbol{a}_n \boldsymbol{x} + \boldsymbol{b}_n) = G(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^d.$$

- The limit G necessarily has GEV margins $G_1, \ldots, G_d$ and admits the representation

$$G(\boldsymbol{x}) = \exp\{-\ell\big(-\log G_1(x_1), \ldots, -\log G_d(x_d)\big)\}.$$

where ℓ is the stable tail dependence function.

Extremal Vulnerability (Theoretical Guarantees)

- All asymptotic considerations for matrix-valued quantities are carried out in $(\mathbb{R}^{d\times d}, \|\cdot\|_F)$, where $\|\boldsymbol{A}\|_F = \{\sum_{i=1}^{d}\sum_{j=1}^{d} a_{i,j}^2\}^{1/2}$ is the Frobenius norm.
- Since d is fixed, convergence in Frobenius norm is equivalent to entrywise convergence.

Theorem (Consistency of XVRank)

Let $\boldsymbol{Y} \sim F$ where $F \in D^d(G)$ is a joint distribution with continuous marginal distribution functions $F_1, \ldots, F_d$, and suppose that $k/\log\log n \to \infty$, as $n \to \infty$. Then it holds that, as $n \to \infty$:

a) $\|\widehat{\boldsymbol{A}} - \boldsymbol{A}\|_F \underset{\text{a.s.}}{\to} 0.$

b) $\|\widehat{\boldsymbol{P}} - \boldsymbol{P}\|_F \underset{\text{a.s.}}{\to} 0.$

c) $\|\widehat{\boldsymbol{\pi}} - \boldsymbol{\pi}\| \underset{\text{a.s.}}{\to} 0.$

Extremal Vulnerability (Theoretical Guarantees)

- The assumption $k/\log\log n \to \infty$ is standard.[3]
- Whereas the intermediate sequence condition ensures that the number of exceedances increases with n, while remaining negligible relative to it, the additional requirement

$$k/\log\log n \to \infty$$

guarantees that k increases sufficiently fast—excluding growth at a rate comparable to $\log\log n$.

- Next, we move to asymptotic normality; below, $\text{vec}(\boldsymbol{B})$ denotes the vectorization mapping,

$$\text{vec}(\boldsymbol{B}) = (b_{1,1}, \ldots, b_{m,1}, b_{1,2}, \ldots, b_{m,2}, \ldots, b_{1,n}, \ldots, b_{m,n})^{\mathrm{T}},$$

where $\boldsymbol{B} = (b_{i,j})$ is a $n \times n$ matrix.

[3] Schmidt, R. & Stadtmuller, U. (2006). Non-Parametric Estimation of Tail Dependence. en. *Scandinavian Journal of Statistics* **33,** 307–335. ISSN: 0303-6898, 1467-9469. (2016).

Extremal Vulnerability (Theoretical Guarantees)

- Assume there exists a function $\mathcal{A} : (0, \infty) \to (0, \infty)$ such that $\mathcal{A}(t) \to 0$, as $t \to \infty$, and

$$\lim_{t \to \infty} \frac{a_{i,j} - t\, \bar{C}_{i,j}(1/t, 1/t)}{\mathcal{A}(t)} = \kappa_{i,j}, \tag{4}$$

for some finite constant $\kappa_{i,j}$, and where

$$\bar{C}(u, v) = u + v - 1 + C(1 - u, 1 - v).$$

Extremal Vulnerability (Theoretical Guarantees)

Theorem (Asymptotic normality of XVRank)

Let $\boldsymbol{Y} \sim F$ where $F \in D^d(G)$ is a joint distribution with continuous marginal distribution functions $F_1, \ldots, F_d$, and suppose that the second-order condition in (4) holds with $\sqrt{k}\mathcal{A}(n/k) \to 0$, as $n \to \infty$. Then, as $n \to \infty$:

a) $\sqrt{k}\, vec(\widehat{\boldsymbol{A}} - \boldsymbol{A}) \xrightarrow{\mathrm{d}} N(\boldsymbol{0}, \boldsymbol{\Sigma}_A)$.

b) $\sqrt{k}\, vec(\widehat{\boldsymbol{P}} - \boldsymbol{P}) \xrightarrow{\mathrm{d}} N(\boldsymbol{0}, \boldsymbol{\Sigma}_P)$, *where* $\boldsymbol{\Sigma}_P = \boldsymbol{G}\, \boldsymbol{\Sigma}_A\, \boldsymbol{G}^\top$.

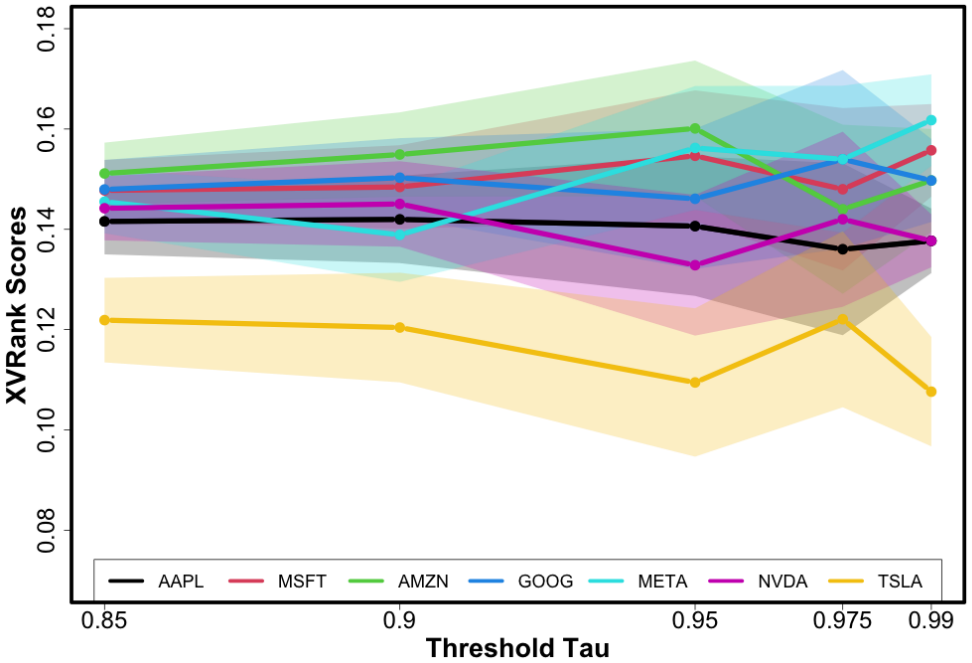
c) $\sqrt{k}\,(\widehat{\boldsymbol{\pi}} - \boldsymbol{\pi}) \xrightarrow{\mathrm{d}} N(\boldsymbol{0}, \boldsymbol{\Sigma})$, *where* $\boldsymbol{\Sigma} = \boldsymbol{H}\, \boldsymbol{\Sigma}_P\, \boldsymbol{H}^\top$.

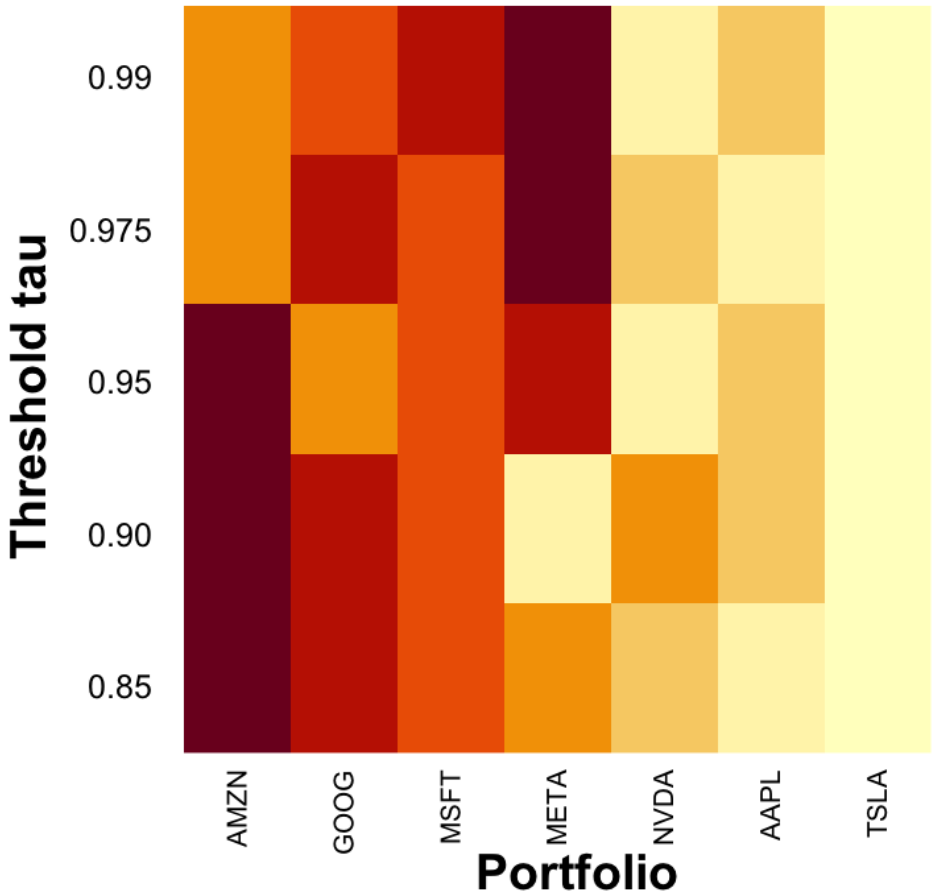
Extremal Vulnerability (Data Application)

- **Magnificent Seven (MAG7) equities:** Apple (AAPL), Microsoft (MSFT), Amazon (AMZN), Alphabet (GOOG), Meta (META), NVIDIA (NVDA), and Tesla (TSLA).
- **Time period:** January 2021 to February 2026.
- **Observations:** Negative (asymmetric) GARCH filtered log returns of daily closing prices.

Extremal Vulnerability (Data Application)

XVRank Scores for MAG7 under several choices for threshold τ

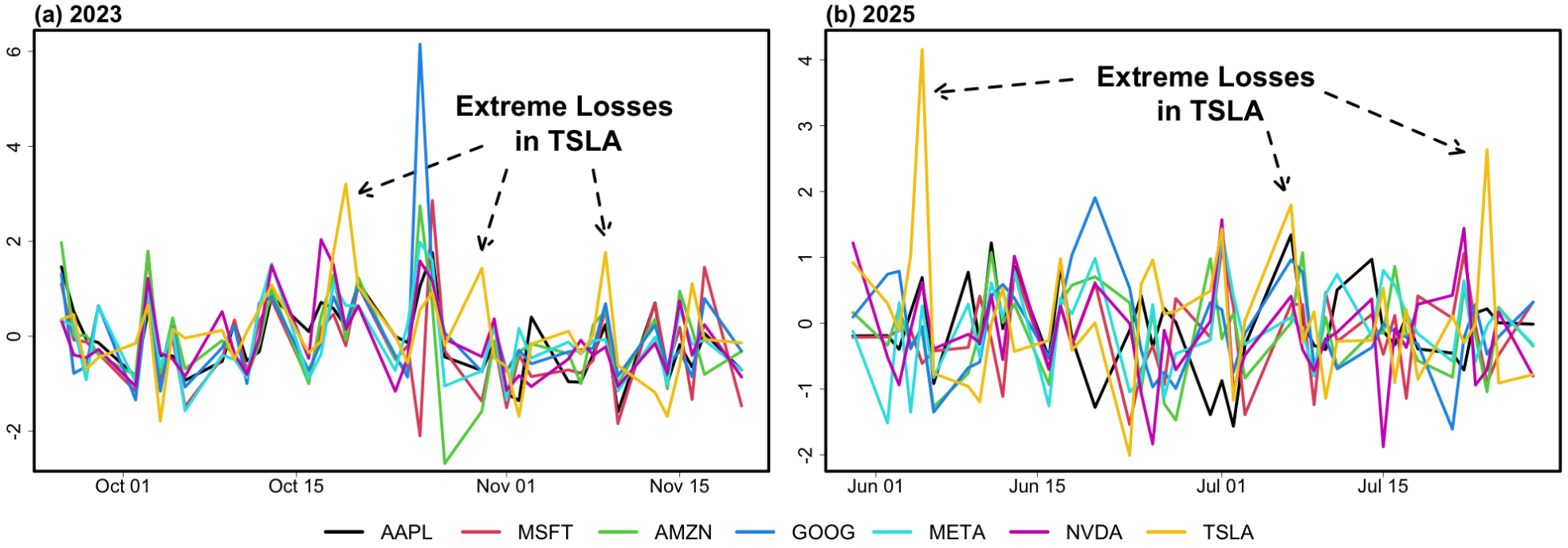




- TSLA is the least vulnerable w.r.t. extreme losses.

Extremal Vulnerability (Data Application)

Implication: This does not mean TSLA does not experience extreme losses. This means extreme losses of TSLA are mostly "unaffected" by the other equities.



Extremal Vulnerability (Data Application)

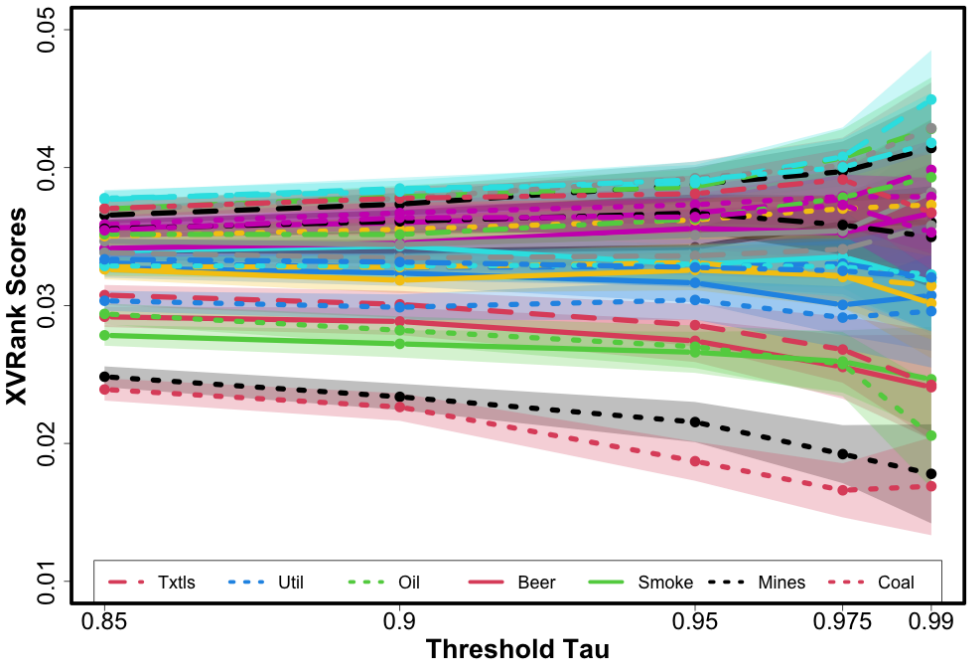
- **Thirty US industry portfolios:** Food, Beer, Smoke, Games, Books, Household (Hshld), Clothes (Clths), Health (Hlth), Chemicals (Chems), Textiles (Txtls), Construction (Cnstr), Steel, Fabricated Products (FabPr), Electrical Equipment (ElcEq), Automobiles (Autos), Transportation Equipment (Carry), Mining (Mines), Coal, Oil, Utilities (Util), Telecommunications (Telcm), Services (Servs), Business Equipment (BusEq), Paper, Transportation (Trans), Wholesale (Whlsl), Retail (Rtail), Meals, Financials (Fin), and Other.

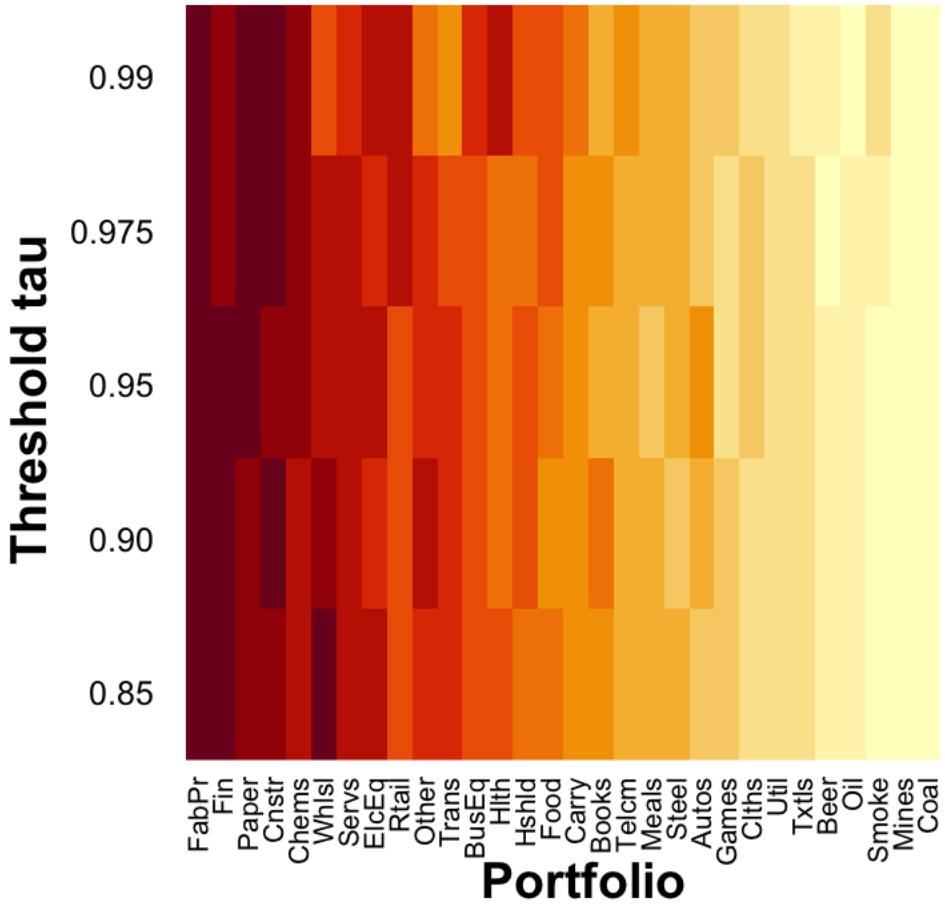
- **Time period:** January 1970 to November 2023.

- **Observations:** Negative (asymmetric) GARCH filtered value-averaged daily returns.

Extremal Vulnerability (Data Application)

XVRank Scores for US industry portfolios under several choices for threshold τ





Extremal Vulnerability (Data Application)

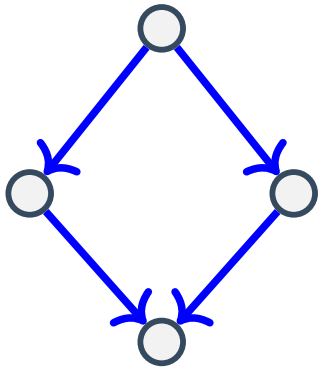
Portfolio	Ave. Rank	Portfolio	Ave. Rank	Portfolio	Ave. Rank
FabPr	**1.61**	Trans	10.68	Autos	19.54
Fin	**2.57**	BusEq	10.79	Games	21.66
Paper	**3.16**	Hlth	12.28	Clths	21.68
Cnstr	**3.17**	Hshld	12.87	Util	23.58
Chems	**5.01**	Food	13.91	Txtls	24.04
Whlsl	6.02	Carry	15.41	**Beer**	**25.79**
Servs	7.23	Books	16.69	**Oil**	**26.13**
ElcEq	8.01	Telcm	18.31	**Smoke**	**26.14**
Rtail	9.51	Meals	18.38	**Mines**	**28.03**
Other	10.29	Steel	18.50	**Coal**	**29.00**

Outline

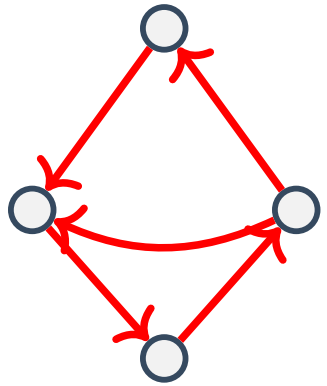
1. Deep learning / Granger Causality
   with Malik Shahid Sultan
2. Extremal Vulnerability
   with Paolo Redondo
3. Topological Effective Connectivity Modeling
   with Anass El-Yaagoubi

Motivation

- Common models of directed dependence assume a DAG structure.
- The brain relies on feedback loops; removing cycles discards physiology.
- **Goal:** detect **what changes, and where**, when a stroke disrupts the network.



DAG



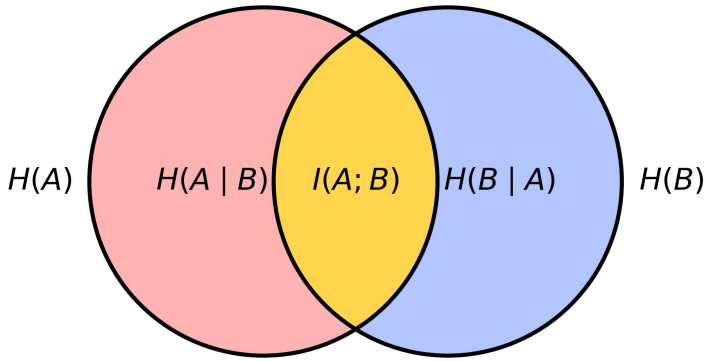
brain: feedback

Measure of Dependence

For channels p and q, sum the lead-lag mutual information over lags:

$$W_{p\to q} = \sum_{k=1}^{K} I\big(X_p(t-k)\,;\, X_q(t)\big).$$

- Direction comes from the time offset: past of p vs. present of q.
- The decomposition below uses the **net flow** on each edge.

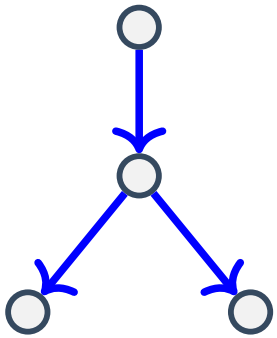


Overlap = mutual information $I(A;B)$.

Hodge Decomposition

The net flow splits into three **orthogonal** components:

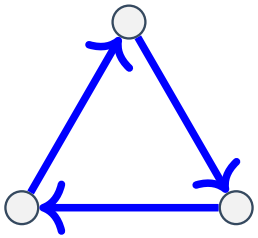
$$W_a = \underbrace{-\delta_0 \varphi}_{\textbf{gradient}} + \underbrace{\delta_1^* \psi}_{\textbf{curl}} + \underbrace{h}_{\textbf{harmonic}}$$



Gradient

feed-forward hierarchy

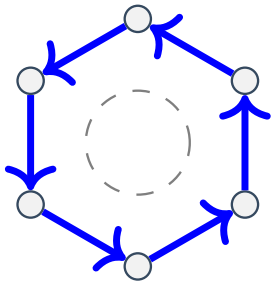
(sources to sinks)



Curl

local feedback

(triangle loops)



Harmonic

global cycles

(flow around holes)

Interpretation of the Components

- **Node potentials φ:** rank regions from sources to sinks.
- **Triangle circulations ψ:** direction and strength of each local feedback loop.
- **Harmonic part h:** residual, longer-range circulation.

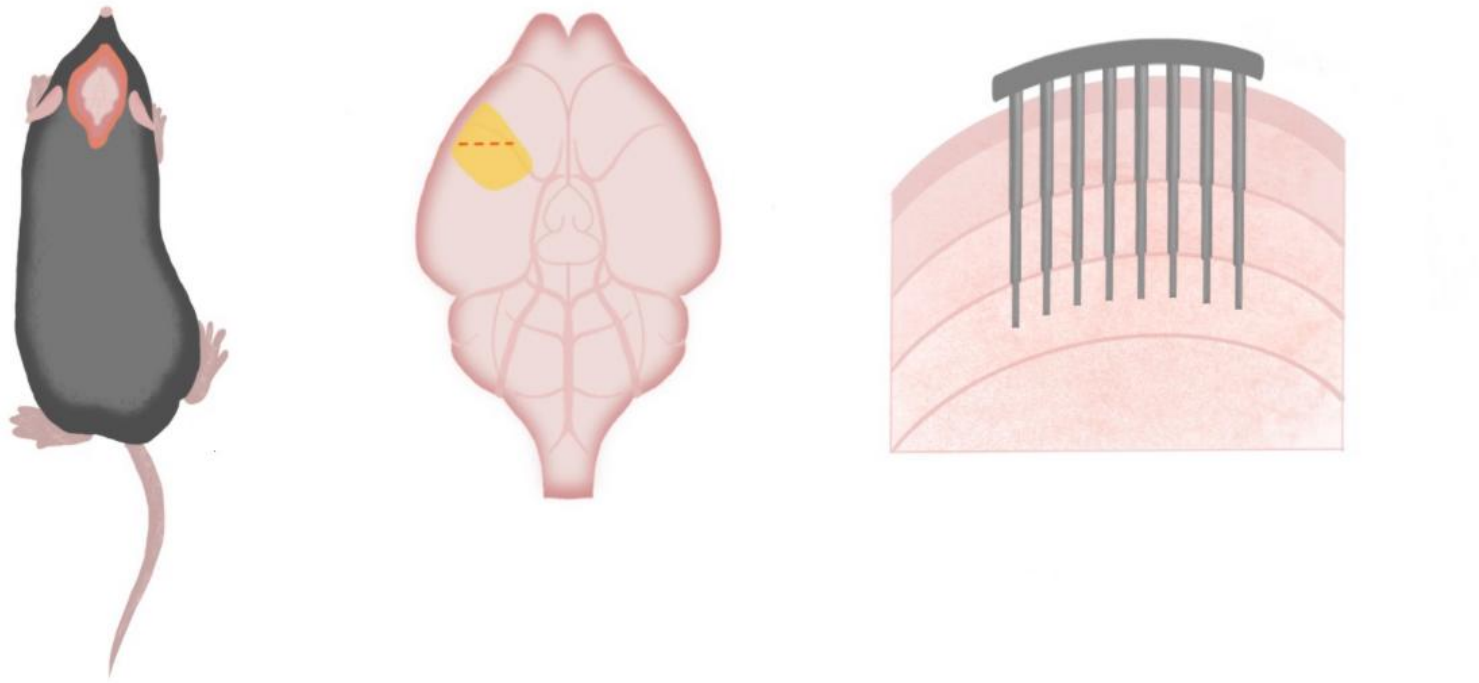
Testing for Changes Between Conditions

Estimate the decomposition per epoch, then compare conditions with standardized mean differences:

$$T_p^{\varphi} = \frac{\bar{\varphi}_p^{\text{post}} - \bar{\varphi}_p^{\text{pre}}}{s_p^{\varphi}}, \qquad T_\tau^{\psi} = \frac{\bar{\psi}_\tau^{\text{post}} - \bar{\psi}_\tau^{\text{pre}}}{s_\tau^{\psi}}.$$

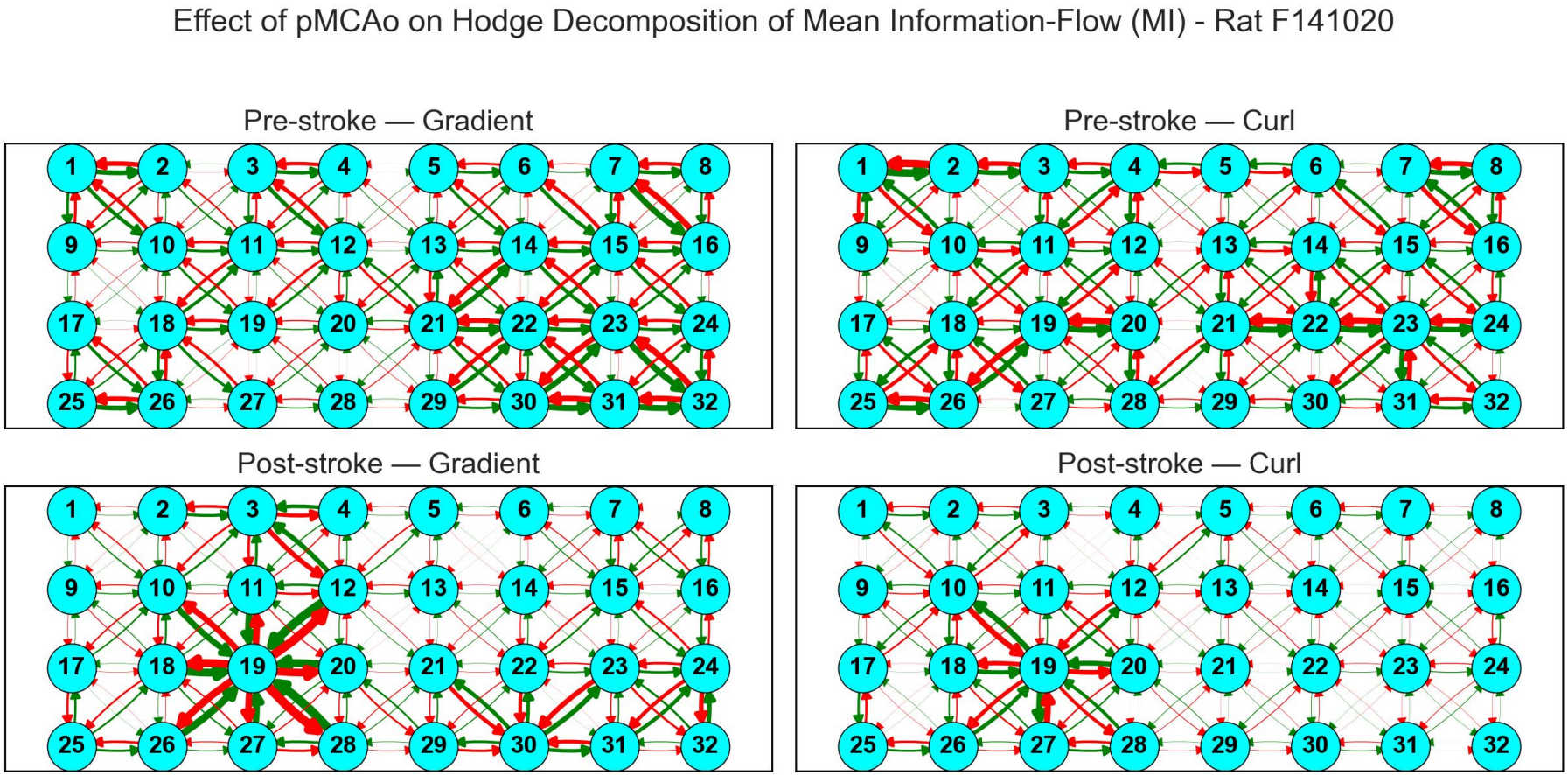
- Null distribution by block permutation of epoch labels (accounts for temporal dependence).
- Benjamini–Hochberg FDR control at $\alpha = 0.05$.
- In simulations, a planted perturbation is detected with no false positives (500 null trials).

Application: Ischemic Stroke in Rat Cortex

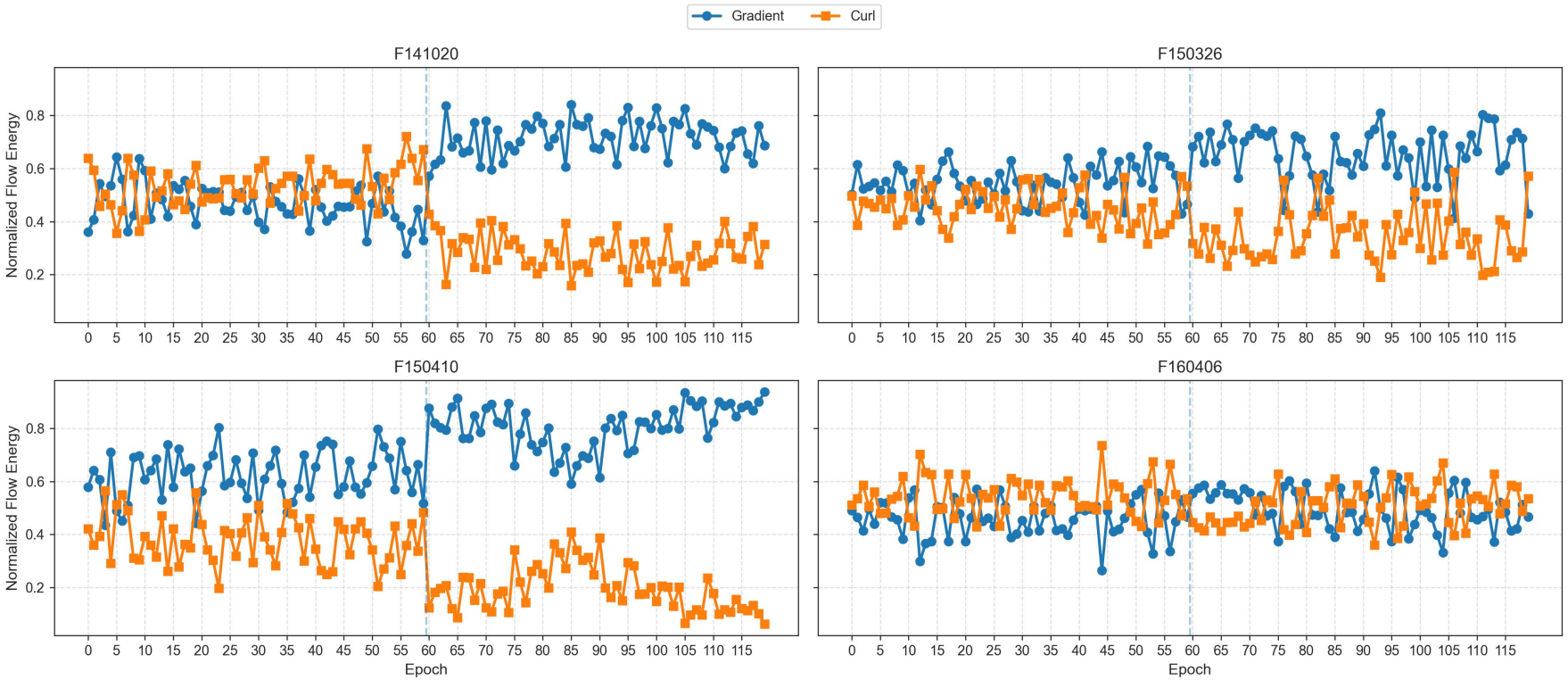


- Induced stroke (pMCAo); 32-channel LFP array (8 sites $\times$ 4 depths): **32 nodes, 94 edges, 84 triangles**.
- **60 pre- vs. 60 post-stroke epochs** per animal (1 s each, 250 Hz).

Estimated Components, Pre- vs. Post-Stroke

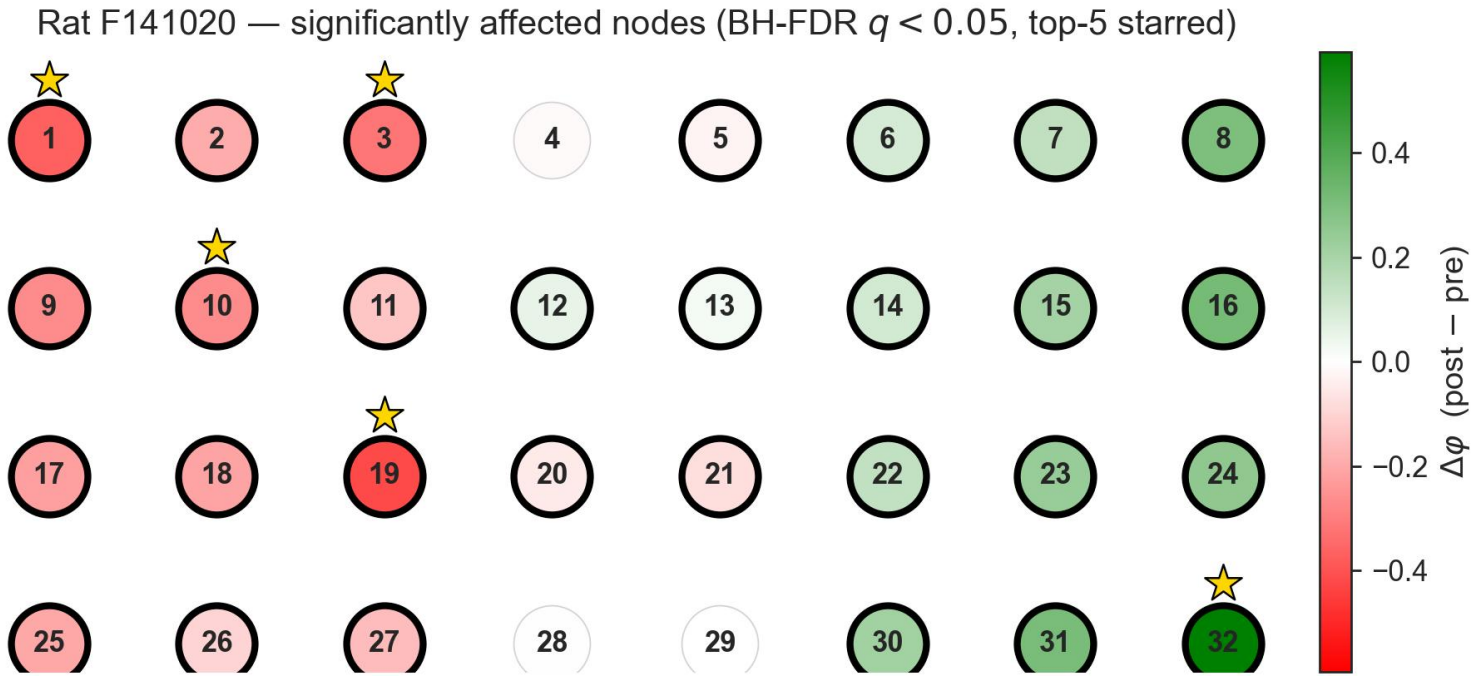


Main Finding: From Feedback to Hierarchy



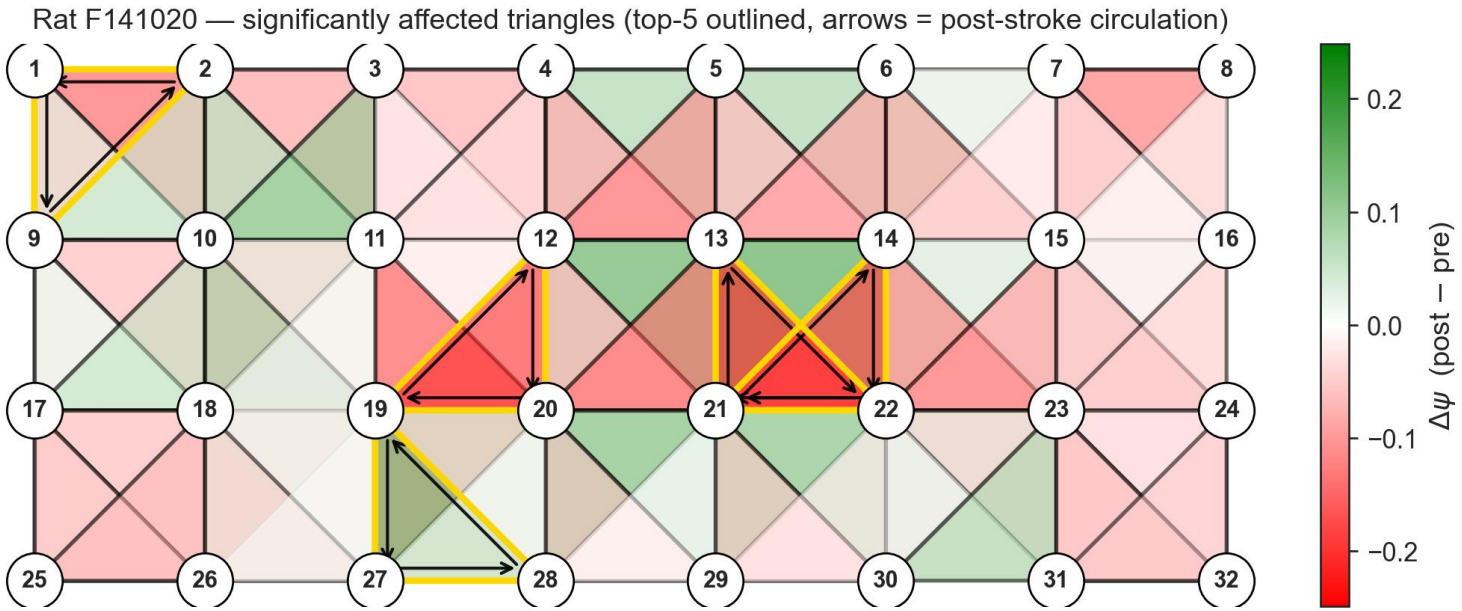
- After stroke: loss of feedback, shift toward **source-driven propagation**.
- Consistent in 3 of 4 animals; one shows no change.

Where Does the Reorganization Happen? Nodes



- **Nodes:** 29 of 32 significant ($q < 0.05$); node 19 becomes a dominant sink, node 32 a dominant source.

Where Does the Reorganization Happen? Triangles



- **Loops:** 66 of 84 triangles significant.

Concluding Remarks

- TECM separates hierarchical drive from recurrent feedback, and tests where directed networks reorganize.
- The framework applies to any directed network with feedback.
- **Future directions:** group-level inference (mixed-effects models); interventional designs toward causal claims.

Thank you! ☺