Deep Learning Without Neural Networks: Fractal-nets for Rare Event Modeling

Yi Huang[1], **James Evans**[1,2] **and Ishanu Chattopadhyay**[1,★]

[1]University of Chicago, Chicago, IL 60637, USA
[2]Santa Fe Institute, Santa Fe NM 87501, USA

★To whom correspondence should be addressed: e-mail: ishanu@uchicago.edu

Complex phenomena of societal interest such as weather, seismic activity and urban crime, are often punctuated by rare and extreme events[1], which are difficult to model and predict. Evidence of long-range persistence[2,3] of such events has underscored the need to learn deep stochastic structures in data for effective forecasts. Recently neural networks (NN) have emerged as a defacto standard for deep learning[4–9]. However, key problems remain with NN inference, including a high sample complexity, a general lack of transparency, and a limited ability to directly model stochastic phenomena. In this study we suggest that deep learning and the NN paradigm are conceptually distinct – and that it is possible to learn "deep" associations without invoking the ubiquitous NN strategy of global optimization via back-propagation[10]. We show that deep learning of stochastic phenomena is related to uncovering the emergent self-similarities in data, which avoids the NN pitfalls offering crucial insights into underlying mechanisms. Using the `Fractal Net` (FN) architecture introduced here, we actionably forecast various categories of rare weather and seismic events, and property and violent crimes in major US cities. Compared to carefully tuned NNs, we boost recall at `90%` precision by `161.9%` for extreme weather events, `191.3%` for light-to-severe seismic events with magnitudes above the local third quartile, and `50.8%` - `418.6%` for urban crime, demonstrating applicability in diverse systems of societal interest. This study opens the door to precise prediction of rare events in spatio-temporal phenomena, adding a new tool to the data science revolution.

COMPLEX systems exhibit rare and sudden transitions that impact dynamical fate[1]. Pre-empting these events can help mitigate crises ranging from catastrophic environmental disasters, to social unrest, market crashes, economic drawdowns on national scales, and global pandemics. However, modeling rare events is challenging: analytical solution from first principle equations, even when known, are seldom tractable, and uncertainties from unresolved scales makes long-range models of the average dynamical behavior incompatible with reliable prediction of low frequency events[11,12]. Here we develop a framework for tracking rare events in the coupled evolution of discrete time stochastic processes. A rare event in our setting is defined to have an occurrence frequency $< 10\%$[13]; extreme weather in US, seismic events around the globe, and violent urban crime in major US cities – all fall in this category. Recent studies in geography[2,3], hydrology[14,15], climate[16,17], and finance[18] suggest rare events might not be completely random, but show long-range persistence[19]. Thus, uncovering deep predictive structures in data is crucial for reliable forecasts[1,20].

Neural Networks (NN) are now a defacto standard for deep learning[4–9,20]. The depth of the NN used to discover prediction-relevant features suggested and sustained the deep metaphor. However, for complex systems in the physical and social sciences, often the key missing piece is effective modeling of stochastic processes. And the fantastic performance of NNs in learning input-output deterministic functions[21–23] carries over imperfectly and with wasted expressive capacity to stochastic phenomena. In the context of low frequency seismic events and extreme weather and crime, we care about precise event localization, underlining the importance of explicit stochastic modeling.

To address this gap we prompt the reader to think of deep learning and the NN paradigm as conceptually distinct — we show that it is possible to learn "deep" associations without invoking the ubiquitous NN strategy of global optimization via backpropagation of the loss gradient[10]. Our key insight is tied to self-similar structures arising in ergodic stochastic processes which take values over a finite alphabet. If such a process has a well-defined set of dynamical states, self-similarity arises naturally: once a trajectory visits a particular state, we

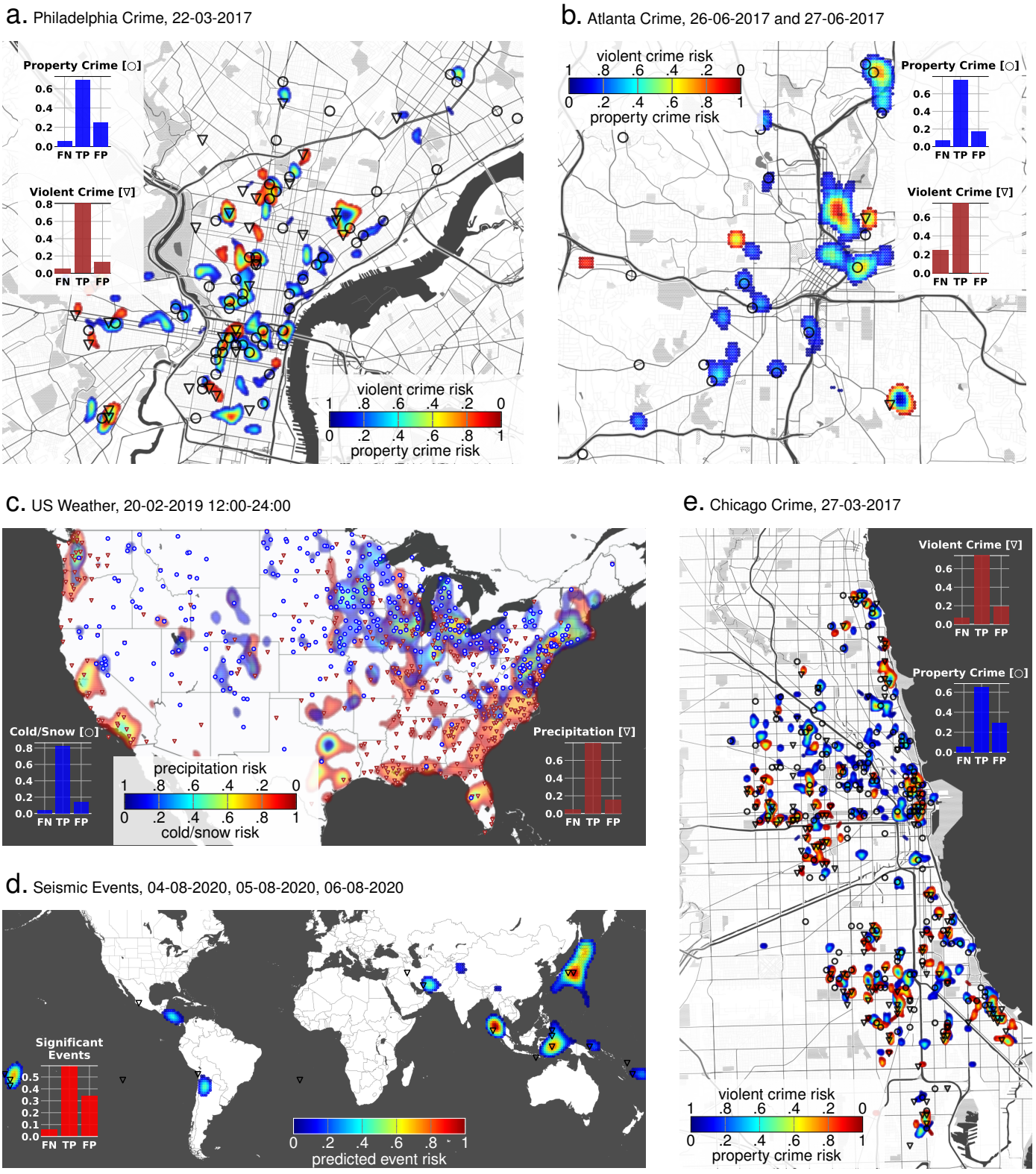


Fig. 1. **Snapshots of Spatio-temporal Rare Event Prediction.** Panels **a-e** illustrate forecasts in diverse systems using `Fractal Nets`, each with distinct spatio-temporal quantization and event definitions (See Extended Data Tab. 1 and 2). With the exception of seismic prediction, these forecasts are actionable, *i.e*, are done early enough (7 days for weather, 3-7 days for urban crime and 120 days for earthquakes), and the events are crucial enough, to trigger mitigation responses. We aim to predict seismic events above the local third quartile magnitude, flagging events with magnitudes > 4.7 on average (≈ 4.5 near Los Angeles, USA) without discriminating between light (< 5.0) or more severe events. Nevertheless we correctly predict 13 out of the largest 15 events in our out-of-sample period (August 2019 - Aug 2020) 120 ± 3 days in advance (See Extended Data Fig. 1).

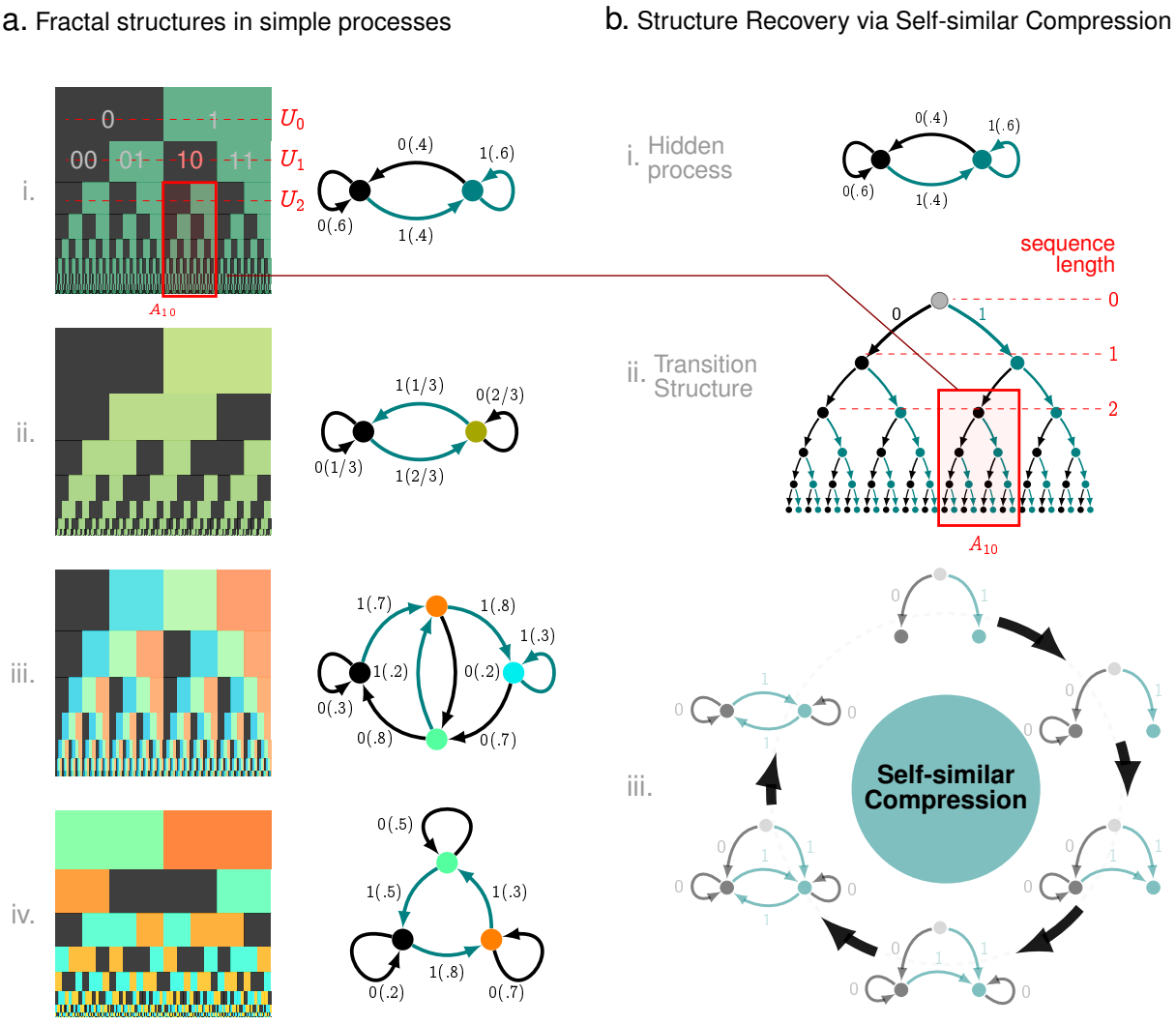


Fig. 2. **Fractal Structures in Stochastic processes and Self-Similar Compression. Panel a.** We map the process states to the unit interval, associating the history of observations from stationary distribution to the binary representation of the points in $[0, 1]$. Stacking these representations as the length of the observation increases reveals process-specific fractal structures shown in (i)-(iv) of panel a. As an example in (i), because every sequence with the last symbol 0 leads to the state q_0, we can interpret U_k as the subset of the unit interval where the k^{th} digit in the binary representation is 0. The block highlighted in red then maps from the sub-tree under the node reached by 10 (and referred to as A_{10}) in the binary tree shown in panel b(ii). Also note that the highlighted block in panel a(i) is a scaled copy of the full representation. **Panel b.** The input data stream induces a m-ary tree, where m is the alphabet size, with probabilities attached to each branch in each split. SSC recursively identifies subtrees that are sufficiently similar to obtain a finite generative model. For example for the the process in panel a(i), we note that the probability of a 0 from the stationary distribution is different from that after we observe a 0, which is also different from after we observe a 1 - leading to producing 2 new states in the first step. But then we realize that the probability of any string after we observe 00 is the same after observing a single 0, implying we have a loop labeled 0 at the state reachable by the first 0. Continuing like this, we distill the complete transition stricture in six steps as shown in panel b(iii).

can "forget" what happened before, implying that future trajectories from each distinct visit of the same state are statistically indistinguishable. Since an ergodic system must revisit its states, the observed dynamical behavior must have self-similar structures, with the set of future paths from any state essentially a scaled copy of the subset of futures from any subsequent visit of the same state. Fig. 2a demonstrates this idea for simple processes with a binary alphabet; we map current states to the unit interval, associating the transpired history with the binary representation of points on $[0, 1]$, revealing the process-specific self-similar organization. Uncovering hidden dynamical states is thus equivalent to recovering fractal structures in observed data, using what we refer to as self-similar compression (SSC) as illustrated in Fig. 2b.

In our approach to learning this emergent self-similar structure, we employ no "neurons", no fixed activation

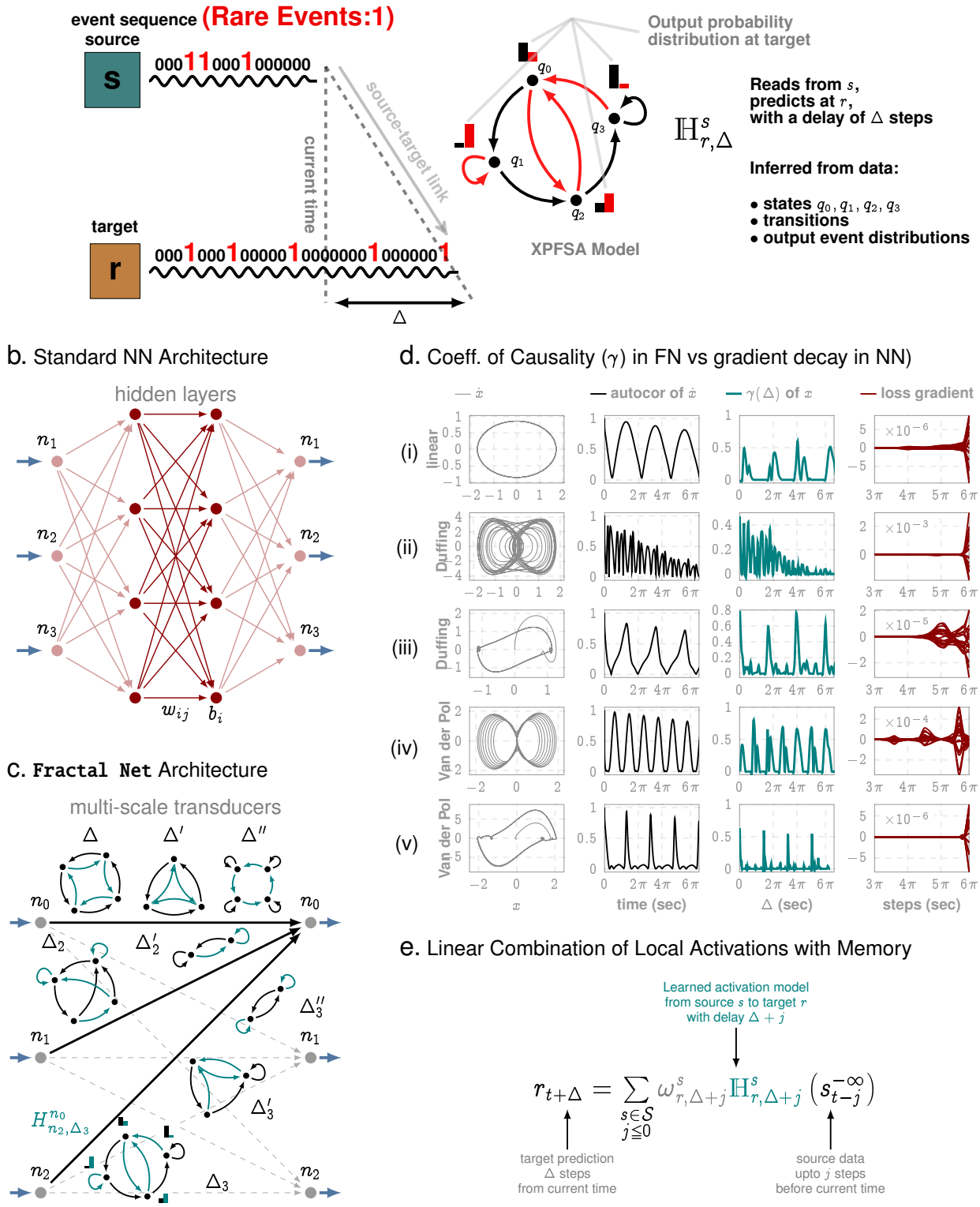
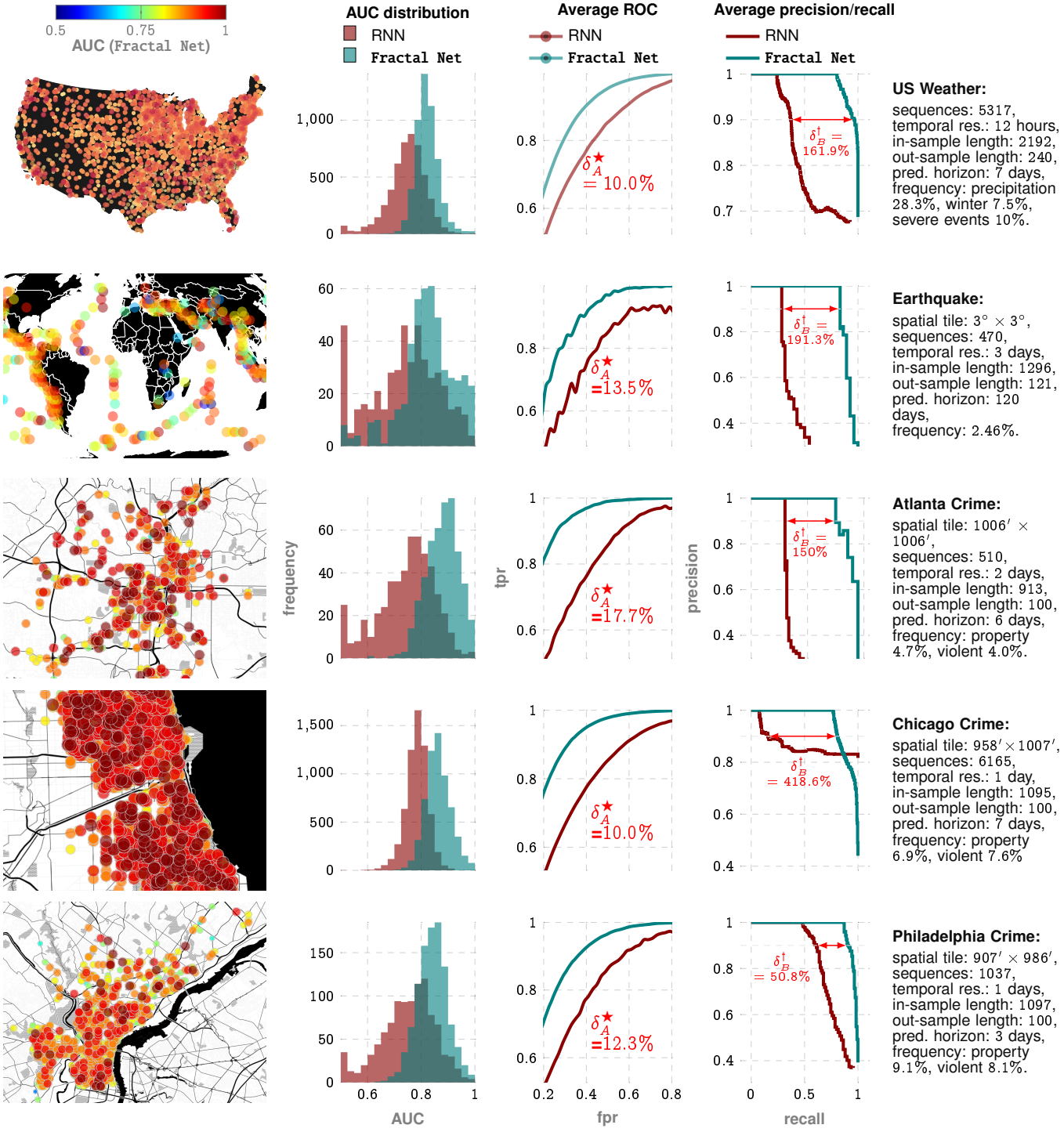


Fig. 3. **Fractal Net Organization**. **Panel a**. SSC computes elemental units, which are crossed probabilistic automata inferred from data, making predictions Δ steps in future of the target stream based on current state estimated from the source stream. **Panels b and c** contrast the schematic differences between the NN and the FN architectures. **Panel d** illustrates the ability to avoid vanishing gradients, where the last column shows the rapidly degrading gradients with Long Short-term Memory (LSTM) recurrent NN models, while column 3 shows the stable behavior of the coefficient of causality γ with time shift Δ. **Panel d** shows the linear combination of local models, highlighting our linear combination of non-linearities instead of non-linear filtering of linear combinations of inputs in NNs

TABLE 1
Performance comparison of the **Fractal Net** prediction and recurrent neural network with LSTM units



★ δ_A: AUC outperformance is measured as the percentage increase in the AUC averaged over the spatial tiles.

† δ_B: Sensitivity outperformance is measured at 90% positive predictive value (PPV).

functions, no user-specified loss functions, and no global optimization via back-propagation. Instead, SSC distills local models (See Fig. 3a), which are assembled into a predictive network (See Fig. 3c) which we call the **Fractal Net** (FN). FN is archetypically distinct from the familiar NN architecture (See Fig. 3b vs Fig. 3c for

contrasting visuals). Each of our elemental units (SSC models) is a probabilistic finite state automaton (PFSA) or a generalized probabilistic automaton for cross-dependencies (crossed PFSA or XPFSA). Thus, given a source s and a target stream r, we infer models $\mathbb{H}^{s}_{r,\Delta}$ (See Fig. 3a), which make predictions Δ steps in future from the current observation step, in the stream r after making observations in the stream s. These models have a finite set of states, with the number of states, the state transition map and the output event probabilities all inferred from data without prior constraints.

Our models are discrete, and they learn from (and predict) categorical input streams. This is appropriately suited for modeling rare event dynamics, where we treat an event as a 1 and its absence 0. In applications with continuous event magnitudes, such as seismic modeling, we use quantization to identify events of interest, $e.g.$, treating all events with magnitude above the local third quartile as a 1.

Instead of the number of layers of neurons, depth in FNs is reflected by average number of states in the inferred SSC models. Instead of assuming fixed memoryless non-linear activation functions (such as tanh, rectified linear unit[10] etc.), here we infer local activation structure from data. The individual SSC models dictate link activation as a function of their current state, which is in turn determined by event history. Number of model states reflects the temporal depth that might be important to determine link state. This results in deeper models with a significantly smaller parameter set (See Extended Data Tab. 2).

Notably, as illustrated in Fig. 3c, each source-target link can have multiple SSC models operating at disparate time scales. Thus, in contrast to layer stacking in Long Short-term Memory (LSTM) NNs, we model multiple timescales explicitly (See Methods: Step Two and Extended Data Algorithm 1 line 11-22). As an added bonus, this approach also addresses the problem of vanishing and exploding gradients. Back-propagation in NNs proceeds by updating network parameters as a function of the iterated loss gradient. Often this decay is too fast for effective learning[24], and despite recent architectural modifications, the problem persists[24,25]. In contrast, SSC inference does not propagate any gradients; identifying a distinct model $\mathbb{H}^{s}_{r,\Delta}$ for each value of Δ. We compare the decay behavior in the two frameworks in Fig. 3d, where 1D systems ranging from being linear (subpanel i) to those operating in non-linear chaotic regimes (subpanel ii-v) are analyzed. We find that the loss gradient vanishes exponentially fast in the case of NNs (column 4) irrespective of the system, whereas for FNs the corresponding measure – the coefficient of causality γ (Column 3) is analogous to the respective autocorrelation functions for $\dot{x}$ (Column 2).

As a final point of architectural contrast, we switch the order of the linear and non-linear operators: the non-linear link activations are combined linearly to be passed on as inputs to downstream nodes (See Fig. 3d), $i.e.$, in FNs we have linear combination of inferred non-linear link activations instead of NNs where we have fixed non-linear activation of linear combination of nodal inputs. Local weights are easily computed with standard regressors, allowing local changes to be integrated on-the-fly (See Methods: Step One).

To briefly describe how SSC actually infers PFSAs or their crossed versions, we note that the input data stream induces a m-ary tree, where m is the alphabet size, with probabilities attached to each branch in each split. SSC recursively identifies subtrees that are sufficiently similar to obtain a finite generative model of the ergodic stationary process (See Fig. 2, panel b, i-iii). Thus SSC uncovers the emergent self-similar structure that arises if different histories lead to identical future stochastic evolution, aiming to find a sufficiently good finite generative model (See Supplementary Text: Defn. 5 and Defn. 7). As in Hidden Markov Models (HMM), we assume to only see outputs from states and not the states themselves. PFSAs are indeed a special class of HMMs (See Supplementary Text: Sec. II-D), but with distinct inference algorithms having the ability to discover structure.

Not all processes have finite generators. The necessary and sufficient condition for our finite models to exist (See Supplementary Text: Sec. II-F and Sec. III) is related to the topological properties of the set of causal states: by definition, we reach the same causal state via distinct paths if the futures are statistically indistinguishable (See Supplementary Text: Sec. I). For SSC to yield a finite model, it is sufficient to have a finite set of causal states, but not necessary (Thm. SI-2). We show that a process has a finite PFSA model only if the set of causal states that are reachable infinitely often ($i.e.$ are persistent) has a finite number of limit points (Thm. SI-5 and Thm. SI-7). As a consequence of these results, we show that NNs are adequate to

model stochastic processes only if every sample path uniquely identifies a single causal state, even if we do not know the initial state precisely, *i.e.*, if all sample paths are *synchronizing inputs*. We show that this condition is equivalent to *a finite set of causal states*, which is the precise criterion for NNs to model stochastic phenomena (Thm. SI-2). Thus, it is now easy to construct counterexamples where NN inference fails irrespective of the number of samples or the number and complexity of layers (See Extended Data Fig. 2).

Finally, precise results on sample complexity for NNs are unknown. In contrast, we establish explicit results on sample complexity, and show that we obtain good models with high probability (See Methods: Performance Analysis and Supplementary Text: Sec. VIII). More precisely, let $\varepsilon, \varepsilon'$ be arbitrarily small positive numbers, with input length that is polynomial in $1/\varepsilon$ and logarithmic in $1/\varepsilon'$, for a reasonably separated family of generating models, the probability that the difference between the inferred and true generating models is bigger than ε as measured by the KL divergence ($\mathcal{D}_{\mathrm{KL}}$) is upper-bounded by ε'. *i.e.*, with input length $n = O\left(poly(1/\varepsilon, \log 1/\varepsilon')\right)$, we have

$$Pr\left(\mathcal{D}_{\mathrm{KL}}\left(\text{generating PFSA} \,\|\, \text{inferred PFSA}\right) > \varepsilon\right) < 1 - \varepsilon' \tag{1}$$

A schematic map of how mathematical development leads to the performance bounds is shown in Extended Data Fig. 3.

To demonstrate FN applicability in diverse spatio-temporal phenomena, we consider 1) rare weather events in contiguous US, 2) global seismic events, and 3-5) urban crime in Atlanta GA, Chicago IL, and Philadelphia PA. These applications, enumerated in Tab. 1 highlight our strictly superior performance over carefully tuned LSTMs. In each of these cases, we begin with a spatio-temporal log enumerating events of interest, along with their space-time coordinates. For example, for US weather, we use events logged by the the Automated Surface Observing Systems (ASOS) network recording extreme precipitation and cold/snow events. The seismic event log is curated by the United States Geological Survey (USGS) hazards program, where we forecast events within $3^\circ \times 3^\circ$ tiles with a magnitude greater than the local third quartile of all events within the past decade. For forecasting urban crime, we log daily occurrences of property crimes (consisting of burglary, theft etc.) and violent crimes (homicide, assault, battery etc.) within a couple of city blocks. We tune discrete time-steps automatically to maximize the average entropy rate of the data streams, resulting in steps measuring 12 hours for the weather models, 3 days for the seismic prediction, and $1-2$ days for urban crime. With the exception of the precipitation event in weather modeling, all event frequencies are lower than 10%. We also choose how far into future we make predictions (prediction horizon), which is chosen to be 1 week for weather, 4 months for seismic prediction, and $3-7$ days for urban crime (See Tab. 1 for details) – performance modestly improves for shorter and degrades rapidly for longer horizons. All these predictions, except perhaps in the case of the seismic events, are actionable, *i.e.*, are done precisely and early enough to intervene or mitigate. For the seismic case, we target events with magnitude greater than the third quartile of the magnitudes of all local recorded events in the past decade. (See Extended Data Fig. 1a, mean: 4.7). Thus, we do not discriminate between light (4-4.9) or more severe events to maintain statistical power. Nevertheless we correctly predict 13 out of the largest 15 events in our out-of-sample period (August 2019 - Aug 2020) 120 ± 3 days in advance (See Extended Data Fig. 1c-d). In all these problems, we significantly outperform carefully tuned LSTMs. As shown in Table 1, we boost sensitivity at 90% positive predictive value by 161.9% for extreme weather events, 191.3% for seismic activity over the local third quartile, 150.0%, 418.6%, and 50.8% for criminal infractions in Chicago, Philadelphia and Atlanta respectively. Outperformance measured by the increased area under the receiver operating characteristic curve for the corresponding problems is given by 10.0%, 9.2%, 17.7%, 10.0%, and 12.3% respectively.

Beyond predictive performance, FNs provide insight into dynamical properties, which is generally difficult with NNs. Each SSC model $\mathbb{H}^s_{r,\Delta}$ has a coefficient of causality $\gamma^s_r(\Delta)$ intuitively defined as:

$$\gamma^s_{r,\Delta} = 1 - \frac{\text{uncertainty of the output } \Delta \text{ steps in future in } r \text{ with observation of the past in } s}{\text{average uncertainty of the output } \Delta \text{ steps in future in } r} \tag{2}$$

specifying how much information about the Δ-step future of the target stream is obtained, per unit bit acquired about the source stream s. As shown in Fig. 3d, for 1D systems γ^s_s is analogous to autocorrelation: this is expected since higher autocorrelation implies higher predictability. In general, the average Δ-dependence of γ^s_s provides us with simple sanity checks. For example, the physical nature of weather and seismic systems

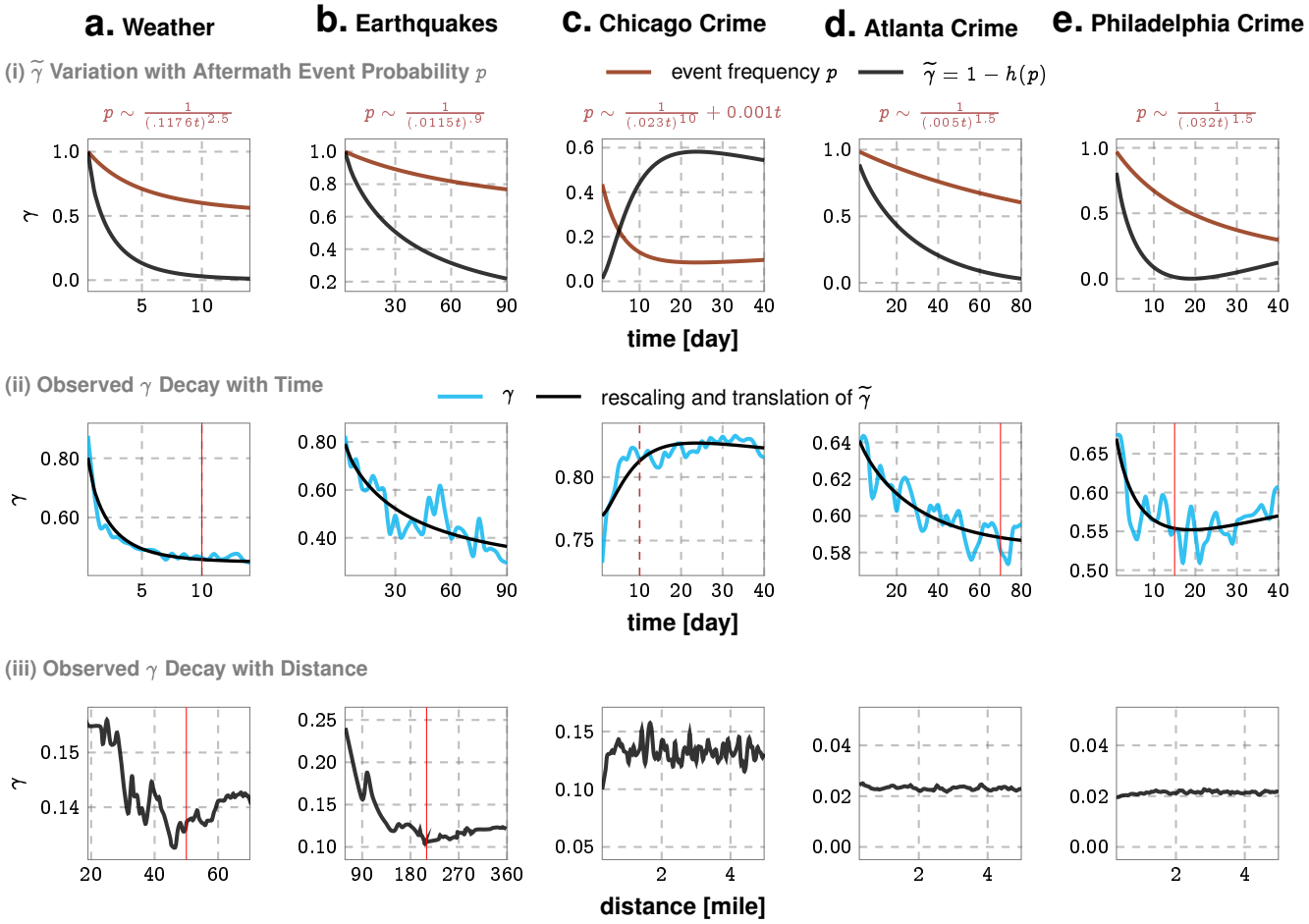


Fig. 4. **Dynamical Properties Revealed by the Coefficient of Causality γ.** Row (i) illustrates that specific decay behaviors for the aftermath event frequency yields different behaviors of $\widetilde{\gamma}$ with time, which is then shown to be consistent with observed $\gamma - \Delta$ behavior (upto scaling and translation) in row (ii). For seismic activity, the aftermath event frequency is known to follow the Omori-Utsu law with a decay exponent (0.7-1.5) consistent with our analysis (1.5). The very different behavior in the case of crime in Chicago is also explainable via rapid decay and subsequent recovery of event frequency as shown in column **c** (See Methods for discussion). Row (iii) illustrates the variation of γ with distance. As required in physical systems (columns 1 and 2) we find a rapid decay. This is absent in the social systems (columns 3-5), suggesting long range persistence in organization akin to critical phenomena.

necessitates an influence decay as we move away in space and time from the event epicenters – no "teleportation" of influence should be possible. Thus, in the neighborhood of events we expect that on average, γ_s^s must decay with increasing Δ, and γ_r^s should decay as physical distance between the source and the target increases. Fig. 4 rows 2 and 3 illustrate that these patterns are correctly recovered for the weather and the seismic systems. Interestingly while the temporal decay also holds true for urban crime, we find no such decay in the spatial dimension for these social systems. This discrepancy possibly suggests that urban spaces operate as one single unit with long-ranged coherence not unlike self-organized criticality suspected to emerge in flocks of birds[26] and other physical systems near criticality.

In addition, $\gamma_s^s(\Delta)$ also sheds light on event frequency after a rare event. Our models have specific states which have large event likelihoods. As we move away from these states, the event likelihood decays. In the context of earthquakes, the increased event frequency (aftershocks) following an event is known to rapidly decay with time according to the empirical Omori-Utsu law[27–32]. This aftermath decay may be related to the $\gamma_s^s(\Delta)$ response. To see this, note that the expression for γ (Supplementary Text Defn. 38) implies:

$$\gamma_r^s = \sum_{i \in Q} \wp_i \left(1 - \frac{h(p^i)}{h(p_0)}\right) \tag{3}$$

where $h(\cdot)$ is the binary entropy function ($h(x) \triangleq x\log(x) + (1-x)\log(1-x)$), and $\wp_i$ is the stationary probability

of the inferred state in stream s, p^i is the event probability in stream r, and p_o is the average event probability. Considering "self-models", *i.e.*, where s and r are the same, we have the bound:

$$\gamma_s^s(\Delta) \leqq a(\Delta) + (1 - h(p_\Delta^E)) \tag{4}$$

where the subscript E refers to state(s) with the maximum event probability, and we use the fact that $\forall i \wp_i \in [0, 1]$ with $\sum_i \wp_i = 1$ and $h(p_0) \leqq 1$. Under the idealized assumption that events are unlikely from states other than E, $a(\Delta)$ is a weak function of Δ, implying that the inferred γ vs Δ plots suggest how the event frequency varies in the aftermath. Fig. 4 illustrates that observed γ behaviors may be approximately reconstructed from variations in the aftermath-event frequency. None of the above insights are possible within the NN framework, in which the gradient decay is purely an artifact of the optimization algorithm, and does not reflect system properties.

A key limitation of FNs is the need for categorical data in self-similar compression. In systems with continuous-valued observations, we can set a magnitude threshold effectively defining the events of interest (as demonstrated in seismic modeling). However more complex event definitions might be warranted elsewhere. Future research will investigate these issues, and attempt to address event frequencies significantly lower to what have been demonstrated here.

Thus, in this study we have laid the groundwork to broaden the applicability of data driven analytics to rare event modeling in complex systems. We hope that this technology, integrated with existing tools, will push the boundaries on our current limits of predictive mitigation of natural disasters and catastrophic societal events.

REFERENCES

[1] Sornette, D. *Why stock markets crash: critical events in complex financial systems*, vol. 49 (Princeton University Press, 2017).

[2] Telesca, L., Cuomo, V., Lapenna, V. & Macchiato, M. Detrended fluctuation analysis of the spatial variability of the temporal distribution of southern california seismicity. *Chaos, Solitons & Fractals* **21**, 335–342 (2004).

[3] Yakovlev, G., Turcotte, D. L., Rundle, J. B. & Rundle, P. B. Simulation-based distributions of earthquake recurrence times on the san andreas fault system. *Bulletin of the Seismological Society of America* **96**, 1995–2007 (2006).

[4] Grefenstette, E., Hermann, K. M., Suleyman, M. & Blunsom, P. Learning to transduce with unbounded memory. In *Advances in neural information processing systems*, 1828–1836 (2015).

[5] Kaiser, Ł. & Sutskever, I. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228* (2015).

[6] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J. & Kaiser, Ł. Universal transformers. *arXiv preprint arXiv:1807.03819* (2018).

[7] Voulodimos, A., Doulamis, N., Doulamis, A. & Protopapadakis, E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience* **2018** (2018).

[8] Liang, H., Sun, X., Sun, Y. & Gao, Y. Text feature extraction based on deep learning: a review. *EURASIP journal on wireless communications and networking* **2017**, 1–12 (2017).

[9] Hutson, M. Ai shortcuts speed up simulations by billions of times (2020).

[10] Bishop, C. M. *Pattern recognition and machine learning* (springer, 2006).

[11] Hamill, T. M. & Whitaker, J. S. Probabilistic quantitative precipitation forecasts based on reforecast analogs: Theory and application. *Monthly Weather Review* **134**, 3209–3229 (2006).

[12] Hu, G., Bódai, T. & Lucarini, V. Effects of stochastic parametrization on extreme value statistics. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29**, 083102 (2019).

[13] Murphy, A. H. Probabilities, odds, and forecasts of rare events. *Weather and forecasting* **6**, 302–307 (1991).

[14] Ouarda, T. B., Girard, C., Cavadias, G. S. & Bobée, B. Regional flood frequency estimation with canonical correlation analysis. *Journal of Hydrology* **254**, 157–173 (2001).

[15] Kantelhardt, J. W. *et al.* Long-term persistence and multifractality of precipitation and river runoff records. *Journal of Geophysical Research: Atmospheres* **111** (2006).

[16] Lennartz, S., Livina, V., Bunde, A. & Havlin, S. Long-term memory in earthquakes and the distribution of interoccurrence times. *EPL (Europhysics Letters)* **81**, 69001 (2008).

[17] Bódai, T. & Tél, T. Annual variability in a conceptual climate model: Snapshot attractors, hysteresis in extreme events, and climate sensitivity. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **22**, 023110 (2012).

[18] Siokis, F. M. Multifractal analysis of stock exchange crashes. *Physica A: Statistical Mechanics and its Applications* **392**, 1164–1171 (2013).

[19] Zhao, X., Shang, P. & Lin, A. Universal and non-universal properties of recurrence intervals of rare events. *Physica A: Statistical Mechanics and its Applications* **448**, 132–143 (2016).

[20] Qi, D. & Majda, A. J. Using machine learning to predict extreme events in complex systems. *Proceedings of the National Academy of Sciences* **117**, 52–59 (2020).

[21] Cybenko, G. Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems* **2**, 303–314 (1989).

[22] Park, J. & Sandberg, I. W. Universal approximation using radial-basis-function networks. *Neural computation* **3**, 246–257 (1991).

[23] Ismailov, V. E. Approximation by neural networks with weights varying on a finite set of directions. *Journal of Mathematical Analysis and Applications* **389**, 72–83 (2012).

[24] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J. *et al.* Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001).

[25] Schmidhuber, J. Learning complex, extended sequences using the principle of history compression. *Neural Computation* **4**, 234–242 (1992).

[26] Mora, T. & Bialek, W. Are biological systems poised at criticality? *Journal of Statistical Physics* **144**, 268–302 (2011).

[27] Omori, F. *On the after-shocks of earthquakes*, vol. 7 (The University, 1894).

[28] Utsu, T. A statistical study on the occurrence of aftershocks. *Geophys. Mag.* **30**, 521–605 (1961).

[29] Enescu, B., Mori, J., Miyazawa, M. & Kano, Y. Omori-utsu law c-values associated with recent moderate earthquakes in japan. *Bulletin of the Seismological Society of America* **99**, 884–891 (2009).

[30] Davidsen, J., Gu, C. & Baiesi, M. Generalized omori–utsu law for aftershock sequences in southern california. *Geophysical Journal International* **201**, 965–978 (2015).

[31] Hainzl, S. & Marsan, D. Dependence of the omori-utsu law parameters on main shock magnitude: Observations and modeling. *Journal of Geophysical Research: Solid Earth* **113** (2008).

[32] GRILLI, L., La MANNA, F. & PACELLI, V. Financial markets, shocks and omori-utsu law. *Journal of Applied Economic Sciences* **13** (2018).

[33] Chattopadhyay, I. & Ray, A. Structural transformations of probabilistic finite state machines. *International Journal of Control* **81**, 820–835 (2008).

[34] Valiant, L. G. A theory of the learnable. *Commun. ACM* **27**, 1134–1142 (1984).

[35] Moosavi, S., Samavatian, M. H., Nandi, A., Parthasarathy, S. & Ramnath, R. Short and long-term pattern discovery over large-scale geo-spatiotemporal data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2905–2913 (2019).

[36] Rnn (2020 (accessed Sep 3, 2020)). URL https://www.tensorflow.org/guide/keras/rnn.

[37] Lstm layers (2020 (accessed Sep 3, 2020)). URL https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM.

[38] Lstm time-distributed layers (2020 (accessed Sep 3, 2020)). URL https://www.tensorflow.org/api_docs/python/tf/keras/layers/TimeDistributed.

[39] Abadi, M. *et al.* TensorFlow: Large-scale machine learning on heterogeneous systems (2015). URL https://www.tensorflow.org/. Software available from tensorflow.org.

[40] Turchetti, C., Conti, M., Crippa, P. & Orcioni, S. On the approximation of stochastic processes by approximate identity neural networks. *IEEE Transactions on Neural Networks* **9**, 1069–1085 (1998).

[41] Goodfellow, I., Bengio, Y. & Courville, A. *Deep learning* (MIT press, 2016).

[42] Sak, H., Senior, A. & Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association* (2014).

MATERIALS AND METHODS

Fractal Net is assembled from local SSC models which are, in general, crossed probabilistic automata (XPFSA). The theoretical development supporting the claims in the main text is presented in the Supplementary Text. A map of how the mathematical proofs relate to each other is presented in Extended Data Fig. 3, which shows that the key concepts (causal states, persistent causal states, synchronization, and accumulation measures) all come together to establish the correctness of the inference algorithm(s). The rest of this section describes the **Fractal Net** construction.

The construction of a **Fractal Net** consists of two steps: 1) local SSC model generation and network pruning and 2) local model aggregation for comprehensive prediction. As discussed in the main text, these local models determine link activation based on their current state, and event prediction is accomplished by aggregating these local activations via a local regressor. No global optimization of these aggregation function is necessary.

The model generation step of **Fractal Net** is accomplished by the algorithms **GenESeSS** (See Extended Data Algorithm 2) and **xGenESeSS** (See Extended Data Algorithm 3). **GenESeSS** and **xGenESeSS** are implementations of the self-similar compression (SSC) discussed in the main text. **GenESeSS** yields PFSA models that capture how the history of an input process influences its own future, and **xGenESeSS** produces XPFSA models that captures how the history of a source process influences the future of a target process. The **Fractal Net** construction is described in Extended Data Algorithm 1, and takes as input a set $\{x_s : s \in S\}$ of length-n time series, hyperparameters ε and $n_0 < n$ for local model inference, $\Delta_{\max}$ for maximum time delay, and γ_0 for thresholding admissible models. For each target sequence x_r, **Fractal Net** outputs a set of admissible models $\mathcal{M}_r$ with a scalar weight for each model in $\mathcal{M}_r$ via model inference and pruning (line 1-10) and training of the aggregation weights (line 11-22).

Step 1: Model inference and pruning

The **Fractal Net** framework models the influence from a source time series x_s on a target time series x_r at a particular time delay Δ by an XPFSA $H^s_{r,\Delta}$ (line 7). Thus, we infer $|S|\Delta_{\max}$ XPFSA models for each x_r which yields $|S|^2\Delta_{\max}$ models in total. Since the number of XPFSA models increases quadratically with the number of time series and strength of the links may vary, pruning low-performing models early is important for parsimony. **Fractal Net** rejects models by thresholding on the *coefficient of causal dependence* $\gamma^s_{r,\Delta}$ of model $H^s_{r,\Delta}$ (line 8), which measures the strength of dependence of the output sequence on the input one. More specifically, we have

$$\gamma^s_{r,\Delta} = 1 - \frac{\text{uncertainty of the next output in } x_r \text{ with observation of } x_s}{\text{uncertainty of the next output in } x_r} \tag{5}$$

γ can be evaluated from the *synchronous composition* (See Supplementary Text Defn. 39) of the PFSA that models the input process (line 6) and the XPFSA that models the causal influence. In Extended Data Fig. 4j we show the synchronous composition of the PFSA in Panel a to the XPFSA in Panel i. For more details on synchronous composition and coefficient of causal dependence, see Supplementary Text Sec. VII. **Fractal Net** retains the model $H^s_{r,\Delta}$ if and only if $\gamma^s_{r,\Delta}$ is greater than a pre-specified threshold γ_0. At the conclusion of Step 1, **Fractal Net** returns an admissible set of models

$$\mathcal{M}_r = \left\{ H^s_{r,\Delta} : \gamma^s_{r,\Delta} > \gamma_0 \right\} \tag{6}$$

for each $r \in S$.

Step 2: Train linear weights

In this step, we integrate the local models in x_r's admissible set for forecasting events in x_r. To do this, **Fractal Net** trains a linear coefficient $\omega^s_{r,\Delta}$ for each $H^s_{r,\Delta} \in \mathcal{M}_r$ (line 22) so that the final prediction for x_r at time step h

is equal to

$$\sum_{H^s_{r,\Delta} \in \mathcal{M}_r} \omega^s_{t,\Delta} H^s_{r,\Delta} \left((x_s)^{h-\Delta} \right), \tag{7}$$

where $(x_s)^{h-\Delta}$ is the truncation of x_s at $h-\Delta$. To compute the coefficients, we solve a regression problem $\mathrm{Reg}(X, \mathbf{y})$ (line 22) for each $r \in S$ with the predictor variables being predictions $\mathbf{x}_t[s, \Delta]$ obtained by running each sequence $(x_s)^{n_0+t-\Delta}$ through $H^s_{r,\Delta}$ (line 17), and the outcome variable being $x_r[n_0+t]$, value of x_r at time n_0+t (line 18). Hence, the X matrix is the $(n-n_0) \times |\mathcal{M}_r|$ matrix with the entry indexed by $t, (s, \Delta)$ given by $\mathbf{x}_t[s, \Delta]$ and $\mathbf{y}$, the $(n-n_0)$-dimensional vector with the entry indexed by t given by $x_r[n_0+t]$. We can solve for the linear weights with any standard regressor.

Performance Analysis of GenESeSS

On line 6 and 7 of Extended Data Algorithm 1, **Fractal Net** calls subroutines **GenESeSS** and **xGenESeSS**. The two algorithms are conceptually similar: while the first infers PFSA as generators of stochastic processes, the second infers XPFSA as models of cross-dependencies between processes. Here, we establish the correctness of **GenESeSS**.

The inference algorithm for PFSA is called **GenESeSS** for Generator Extraction Using Self-similar Semantics. Both the derivation of the PFSA model and its SSC inference are based on the concept of *causal state*. A dynamical system reaches the same causal state via distinct paths if the futures are statistically indistinguishable. More precisely, each process over an alphabet Σ of size m gives rise naturally to an m-ary tree with the nodes at level d being sequences of length d, and the edge from the node x to $x\sigma$, $\sigma \in \Sigma$, labeled by $Pr(\sigma|x)$ – the probability of observing σ as the next output after x. By the definition of causal state, if two subtrees are identical with respect to edge labels, then their roots are sequences that lead the system to the *same* causal state. We show in Supplementary Text Sec. II and III that, for a process of Markov order k, identifying all the roots of identical subtrees indeed offers a *finite* automaton structure whose unique strongly connected component is the generating PFSA of the process. The automaton structure obtained from this subtree "stitching" procedure is conceptualized formally in Defn. 1.

Definition 1 (Probabilistic Finite-State Automaton (PFSA))**.** *A PFSA G is a quadruple $(Q, \Sigma, \delta, \tilde{\pi})$, where Q is a finite set, Σ is a finite alphabet, $\delta : Q \times \Sigma \to \Sigma$ is called the transition map, and $\tilde{\pi} : Q \to \mathbf{P}_\Sigma$, where $\mathbf{P}_\Sigma$ is the space of probability distributions over Σ, is called the transition probability. (Supplementary Text Sec. II)*

Step 2 of the Extended Data Algorithm 2 (line 5-19) is an implementation this subtree "stitching" approach under finiteness of input data. Note that the criterion for "stitching" two subtrees with roots x and x' is that their edge labels are identical for *all* depths, which translates to $p(y|x) = p(y|x')$ for sequence y of all lengths. The criterion is not verifiable with finite data, and hence **GenESeSS** identifies two subtrees if they agree on depth one. Defining *symbolic derivative* ϕ_x to be the vector with the entry indexed by σ given by $p(\sigma|x)$, **GenESeSS** identifies x and x' if $\phi_x = \phi_{x'}$. This approach works well under the assumption that the target PFSA is in *general position*, meaning that different causal states have distinct symbolic derivatives. In practice, **GenESeSS** uses *empirical symbolic derivative* defined below to approximate ϕ_x. Let x be an input sequence of finite length, the empirical symbolic derivative $\hat{\phi}^x_y$ of a sub-sequence y of x is a probability vector with the entry indexed by σ given by

$$\hat{\phi}^x_y(\sigma) = \frac{\text{number of } y\sigma \text{ in } x}{\text{number of } y \text{ in } x} \tag{8}$$

GenESeSS identifies two sequences (line 12) if their *empirical* symbolic derivatives are within an ε-neighborhood of each other for certain $\varepsilon > 0$.

For simplicity, we first illustrate how **GenESeSS** solves the transition structure of the target PFSA from a sample path x generated from a process of Markov order k. Assuming the x_0 produced by Step 1 (line 4) is λ, the empty sequence, **GenESeSS** starts by calculating $\hat{\phi}^x_\lambda$, *i.e.*, the empirical distribution on Σ, and records λ as the identifier of the first state. Then, **GenESeSS** appends λ with each $\sigma \in \Sigma$, and calculates $\hat{\phi}^x_\sigma$. By the general position assumption and assuming x is long enough, with high probability, no $\hat{\phi}^x_\sigma$ is within an ε-neighborhood

of $\hat{\phi}^x_{\sigma'}$ for $\sigma \neq \sigma'$, and hence each σ is recorded as the identifier for a new state. In fact, **GenESeSS** will keep on appending symbols to identifiers of stored states and adding new states until it reaches a sequence of length $k+1$. Assuming $y = \sigma_1 \cdots \sigma_k \sigma_{k+1}$, since the process is of order k, we have $\phi_y = \phi_z$ for $z = \sigma_2 \cdots \sigma_{k+1}$, and hence, with high probability, $\hat{\phi}^x_y$ and $\hat{\phi}^x_z$ can be within an ε-neighborhood of each other given long enough input x. In this case, **GenESeSS** identifies the state represented by y with that of z. In fact, **GenESeSS** will identify all states represented by sequences of length $k+1$ to some previously-stored states. And since no new states can be found, **GenESeSS** exits the loop on line 8 after iteration $k+1$. Taking the strongly connected component on line 19, **GenESeSS** gets the correct transition structure. See Supplementary Text Sec. III for the proof that the PFSA thus inferred indeed generates the process. In Extended Data Fig. 4e and f, we show the labeled directed graph obtained by subtrees stitching for two processes with Markov order 1 and 2, respectively. The two examples are discussed in Supplementary Text Example 1 and 3, and their tables of causal states can be found in Extended Data Tab. 3 and 4. The generating models of the two processes are the PFSA in Extended Data Fig. 4a and b, respectively. We point out that the unique strongly connected component of graph in Panel e is the PFSA in Panel a and and f, b.

However, not all processes generated by PFSA have finite Markov order. For examples, the processes generated by PFSA in Extended Data Fig. 4c and d (See Supplementary Text Examples 4 and 5 for detail) do not have finite Markov order. For such cases, Step 2 of **GenESeSS** will never exit in theory, since there exists no $n \in \mathbb{N}$ such that every causal state is visited for sequences with length $\leq n$. And if we implement an artificial exit criterion, the model inferred might be unnecessarily large, and have hard-to-model approximations. We address this issue via the notion of synchronization – the ability to identify that we are localized or synchronized to a particular state despite being uncertain of the initial state.

The solution for inferring succinct models in non-finite Markov order cases lies in the concept of *synchronization*. In Step 1 of Extended Data Algorithm 2 (line 1-4), **GenESeSS** finds an almost synchronizing sequence, which allows **GenESeSS** to distill a structure that is similar to that of the finite Markov order cases, and thus carry out the subtree "stitching" procedure described before.

A sequence x is *synchronizing* if *all* sequences that end with the suffix x terminates on the same causal state. A process is synchronizable if it has a synchronizing sequence, and a PFSA is *synchronizable* if the process it generates is synchronizable. All processes of finite Markov order, and hence their generating PFSA, are synchronizable.

Example: An example of synchronizable process without finite Markov order is given by the process generated by the PFSA in Extended Data Fig. 4c. The shortest synchronizing sequence of the PFSA is 11. The binary tree of the process generated by the PFSA is given in Panel g, with darker nodes representing synchronizing sequences. See Supplementary Text Example 4 for detail. We show in Supplementary Text Sec. IV that, although a synchronizable process generated by a PFSA may have an infinite set of causal states, only *finitely* many among them are *persistent*, *i.e.* repeated with non-vanishing probabilities with increasing sequence length. We also show in Supplementary Text Sec. III that any process having a finite set of persistent causal states whose sum of probabilities approaches 1 as the sequence length increases has a PFSA generator whose state set Q is in one-to-one correspondence with the set of persistent causal states. In Extended Data Fig. 5a, we show the probability of sequences synchronizing to the three states of the PFSA in Extended Data Fig. 4c. We can see that the sum approaches 1 as the sequence length increases. Hence, although the process has no finite Markov order, it has a PFSA generator.

Since any sequence prefixed by a synchronizing sequence is synchronizing, as long as Step 1 of **GenESeSS** produces a synchronizing sequence, Step 2 of **GenESeSS** will work *solely* with causal states represented by a synchronizing sequence. The analysis above implies that, assuming a process has a PFSA generator, the PFSA obtained by subtree stitching on a *subtree rooted at a synchronizing sequence* is indeed the generating PFSA of the process. We show in Extended Data Fig. 6g the *running tree* of **GenESeSS** for the PFSA in Panel e. A running tree of **GenESeSS** rooted at state q visualizes the run of **GenESeSS**, given that the x_0 found in Step 1 is a synchronizing to state q, and **GenESeSS** correctly identifies all new and repeating states. Nodes colored orange in the tree are the new states **GenESeSS** finds in its run, while the gray states are the repeating ones. Note that, in the running tree, if we let the gray nodes (repeating states) to travel along the gray lines until they

overlaps with the orange nodes (stored states) with a matching label, we get the PFSA in Panel e back.

Finally, we describe the scenario of recovering PFSA from non-synchronizable processes. An example of a non-synchronizable process is given in Extended Data Fig. 4d (See Supplementary Text Example 5 for detailed description). The directed graph obtained by the subtree stitching for sequences up length 5 is shown in Panel h. Recall that **GenESeSS** can recover synchronizable PFSA with the help of synchronizing sequence because of the one-to-one correspondence between the set persistent causal states and the state set of the PFSA. However, as we show in Supplementary Text Sec. III Example 7, the set of persistent causal state of this PFSA is *empty*. In this specific example, the set of causal states has the form $\{q_n : n \in \mathbb{Z}\}$, where $\mathbb{Z}$ is the set of integers. The probability of the causal state q_n for sequence length d, denoted as $p_d(q_n)$, can be calculated from the recursive formula (61) and we plot $p_d(q_n)$ vs n in Extended Data Fig. 5b for $d = 100, 200, 300, 400$. It follows from Eq. (61) and also by direct observation of Fig. 5b that, as the sequence length increases, the curves flatten out, suggesting $\lim_{d\to\infty} p_d(q_n) = 0$ for all n and hence the PFSA has no persistent states.

In order to obtain a finite generator, we consider the topological properties of the set of causal states, specifically the closure of the set (See Supplementary Text Sec. III). We show that if the closure of the causal states has finitely many *atomic* accumulation points, then the process is generated by a PFSA whose set of states is in one-to-one correspondence with the set of atomic accumulation points of the set of causal states. In Extended Data Fig. 5C, we show the cumulative density function of $\phi_x(0)$ (which is the probability of producing 0 after observing x) for $|x| = 20$ and 40. We can see that the curve approaches a step function with *two steps*. In fact, we can show that the two steps correspond to $q_{-\infty}$ and q_{∞}, the two limit points of the set of causal states. See Supplementary Text Example 8 for detail.

From the analysis above, we see that the problem of recovering a finite structure of the target PFSA from non-synchronizable process boils down to approximating atomic accumulation points of the set of causal states, which induces the concept of ε-synchronization[33].

A sequence x is *ε-synchronizing* to the state q if the distribution $\wp_x$ on the state set Q induced by x satisfies $\|\wp_x - \mathbf{e}_q\|_\infty < \varepsilon$, where $\mathbf{e}_q$ is the base vector with 1 on the entry indexed by q and 0 elsewhere. We show in Supplementary Text Sec. IV that there exists ε-synchronizing sequence for arbitrarily small $\varepsilon > 0$. The importance of ε-synchronizing sequence is twofold: 1) since $\phi_x^T = \wp_x^T \widetilde{\Pi}$, where $\widetilde{\Pi}$ is the $|Q| \times |\Sigma|$ matrix with the row indexed by q given by $\widetilde{\pi}(q)$, a $\wp_x$ close to $\mathbf{e}_q$ give rise to a ϕ_x close to $\widetilde{\pi}(q)$. And 2) although sequences prefixed by an ε-synchronizing sequence to a state q may not remain ε-synchronizing to state q, they are close to q *on average*. As an example, let us consider the non-synchronizable PFSA in Extended Data Fig. 6f over an alphabet of size 3. In Extended Data Fig. 6a and c, we show scatter plots of points $(\phi_x(0), \phi_x(1))$ and $(\phi_{0^5x}(0), \phi_{0^5x}(1))$ for $|x| = 3$ and 9. In Panel b, we show the density plot with a Gaussian kernel of points in panel a weighted by $p(x)$ and, in Panel d, points in panel c weighted by $p\left(x|0^5\right)$. Although Panel b shows that, when sequence length gets bigger, the points become more clustered to the red dots, which are $(\widetilde{\pi}(q,0), \widetilde{\pi}(q,1))$ for the three states of the PFSA, Panel a shows that ϕ_xs can be too scattered for **GenESeSS** to derive a succinct model. However, by appending all sequences with 00000, a sequence with induced distribution on states, $\wp_{0^5}$, close to $\mathbf{e}_s$, not only the density plots in Panel d show a much tighter clustering around the red dots, the scatter plots in Panel c also shows points are better clustered to the red dots.

To find an almost synchronizing sequence algorithmically, **GenESeSS** first calculates the convex hull of symbolic derivatives of subsequences of x up to length L (line 1-3), and then selects a sequence x_0 whose symbolic derivative is a vertex of the convex hull (line 4). Since the convex hull of $\left\{\phi_x : x \in \Sigma^L\right\}$ is a linear projection of the convex hull $\left\{\wp_G(x) : x \in \Sigma^L\right\}$ via $\widetilde{\Pi}$, we can expect sequence x with ϕ_x being a vertex of the convex hull of $\left\{\phi_x : x \in \Sigma^L\right\}$ to be a good candidate for an almost synchronizing sequence.

Due to the finiteness of data, we cannot expect to distinguish causal states whose marginals on Σ are too close. Thus, to make explicit our identifiability conditions, we require the transition probability $\widetilde{\pi}$ of the target PFSA to be μ-distinguishable, *i.e.* $\|\widetilde{\pi}(q) - \widetilde{\pi}(q')\|_\infty > \mu \quad \forall q \neq q' \in Q$. Under these conditions, we show in Supplementary Text VIII a *probabably approximately correct (PAC)*[34] learnability for the class of synchronizable μ-separable PFSA with $|Q| \leq M$ and $\min\{\widetilde{\pi}(q,\sigma) : \widetilde{\pi}(q,\sigma) > 0\} \geq \eta$. In particular, assuming $\varepsilon \leq \min\{\mu/2, \eta\}$

and the length L for finding a synchronizing sequence is in the order of $\log_{|\Sigma|} 1/\varepsilon$, then we have

$$Pr\left(\mathcal{D}_{\mathsf{KL}}\left(G \,\|\, G'\right) > \varepsilon\right) < 1 - \varepsilon' \tag{9}$$

where G' is a PFSA inferred from a sequence generated by G and of minimum length n satisfying:

$$n = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\iota}\left(\frac{1}{\eta}\right)^{M+L(\varepsilon)}\right) \tag{10}$$

We can also show that, assuming G is a synchronizable PFSA and the sequence x_0 found in Step 1 is a synchronizing sequence, then an inferred PFSA G' from a length n sequence generated by G satisfies

$$\mathbb{E}\left(\mathcal{D}_{\mathsf{KL}}\left(G' \,\|\, G\right)\right) \leq \frac{1}{n\eta^{|Q|+L}} \tag{11}$$

See Extended Data Fig. 3 for a summary of the theoretical development discussed above for guaranteeing algorithmic performance as a directed graph.

Performance Analysis of xGenESeSS

The inference algorithm for XPFSA is called **xGenESeSS**, which takes as input two sequences x_{in}, x_{out}, and a hyperparameter ε, and outputs an XPFSA in a manner very similar to the inference algorithm of PFSA. See Extended Data Algorithm 3 for detail.

While a PFSA models how the past of a time series influences its own future, a XPFSA models how the past of an input time series influences the future of an output time series. Hence, while in the SSC algorithm of PFSA, we identify sequences if they lead to futures that are statistically indistinguishable, in the SSC algorithm of XPFSA, we identify sequences if they lead to the same future distribution of the *output*.

Definition 2 (Crossed Probabilistic Finite-State Automaton (XPFSA))**.** *A crossed probabilistic finite-state automaton is specified by a quintuple* $(\Sigma_{in}, R, \eta; \Sigma_{out}, \chi)$*, where* Σ_{in} *is a finite input alphabet,* R *is a finite state set,* η *is a partial function from* $R \times \Sigma_{in}$ *to* R *called transition map,* Σ_{out} *is a finite output alphabet, and* χ *is a function from* R *to* $\mathbf{P}_{\Sigma_{out}}$ *called output probability map, where* $\mathbf{P}_{\Sigma_{out}}$ *is the space of probability distributions over* Σ_{out}*. In particular,* $\chi(r, \tau)$ *is the probability of generating* $\tau \in \Sigma_{out}$ *from a state* $r \in R$ *(See Supplementary Text Sec. VII).*

Extended Data Fig. 4i gives an example of an XPFSA with 4 states. Note that a XPFSA has no transition probabilities defined between states as a PFSA does. The XPFSA in the example has a binary input alphabet and an output alphabet of size 3. The bar charts next to the 4 states of the XPFSA indicate the output probability distributions. To generate a sample path, an XPFSA requires an input sequence over its input alphabet.

Similar to the PFSA construction approach, here we compute the *cross symbolic derivative*, which is the ordered tuple $Pr(\tau|x)$, with $\tau \in \Sigma_{\mathsf{out}}$ and a sequence x over Σ_{in}. We compute the empirical approximation of the cross symbolic derivative from sequences x_{in} and x_{out} as:

$$\hat{\phi}_y^{x_{\mathsf{in}}, x_{\mathsf{out}}}(\tau) = \frac{\text{number of } \tau \text{ in } x_{\mathsf{out}} \text{ after } y \text{ transpires in } x_{\mathsf{in}}}{\text{number of sub-sequence } y \text{ in } x_{\mathsf{in}}} \tag{12}$$

Thus, **xGenESeSS** is almost identical to **GenESeSS** except that, in Step 1, **xGenESeSS** finds an almost synchronizing sequence based on cross symbolic derivatives, and in Step 2, identifies the transition structure based on the similarity between cross symbolic derivatives. Arguments for establishing the effectiveness of **GenESeSS** carry over to **xGenESeSS** with empirical symbolic derivative replaced by empirical cross symbolic derivative (See Supplementary Text Defn. 44).

Loss Function As Generalized KL Divergence Between Stochastic Processes

Deviation of deterministic functions may be measured in diverse metrics, leading to the necessity of a user-defined choice of loss functions in NNs. In contrast, the deviation between stochastic processes needs to be

measured in terms of some quantification of the deviation of the associated finite dimensional distributions (FDD). Thus, any notion of a loss that we use must have the property that a zero loss indicates convergent FDDs. We quantify this loss via defining the notion of KL divergence between two PFSA, extending the well-known information-theoretic measure of deviation of probability distributions to stochastic processes.

We show that the log-likelihood of a sequence being generated by a second PFSA converges to the sum entropy rate of the generating PFSA and the KL divergence of the second process from the actual generator (See Extended Data Fig. 5d-g). Performance of the **GenESeSS** is then measured in the term of KL divergence between the actual and inferred processes (See Sec. and Supplementary Text Sec. VIII). It is still possible to have different choices of the loss metric by defining functions of the KL divergence between processes. However, they all produce qualitatively similar results.

Application Details: Data Source, Preprocessing & Performance Comparison

We demonstrate five applications of FN modeling in complex spatio-temporal phenomena: predicting rare weather events in contiguous US, global seismic events with magnitude registering above the local third quartile, and forecasting urban crime in Atlanta GA, Chicago IL, and Philadelphia PA. These applications, enumerated in Tab. 1, 1) highlight the strictly superior performance of FN over LSTMs, 2) underlines the parsimony of the FN models, and 3) provide important insights into the underlying "physics" of the system at hand.

In each of these systems, we begin from a spatio-temporal event log, which enumerates events of interest, along with their space-time coordinates. For example, for US weather, the log is a culmination of events recorded in one of the 2000 airport-based weather stations as a part of the ASOS network logging extreme precipitation and cold/snow events. The seismic event log comes from the USGS hazards program, where we attempt to forecast events with a magnitude which is greater than the local third quartile of all events recorded within the past decade. For urban crime, we take a similar approach logging property crimes (consisting of burglary, theft etc.) and violent crimes (homicide, assault, battery etc.) within a couple of city blocks. In each application, we need to choose a spatial discretization to define our tiles which, for each event type, defines a distinct variable that we model and make predictions on. For example, in the weather prediction problem, we have 2000 spatial tiles, and three event categories (precipitation, cold/snow, and severity magnitude≥ 3[35]), resulting in 6000 variables. We eliminate some tiles which have event frequencies under 1%, leaving us with 5317 variables, or time series sequences. In case of seismic events, we discretize the globe into $3^\circ \times 3^\circ$ tiles and consider events (event magnitude $>$ average event magnitude), where the average is taken over all events recorded in the past decade above magnitude 3.9. As before, we eliminate tiles which are not seismically active leaving us with 470 variables or sequences. In case of urban crime, we follow a similar approach by covering the city with a grid spanning a couple of city blocks, and eliminating tiles which lack sufficient number of events. This leads to 6165 variables in Chicago, 510 sequences in Atlanta and 1037 sequences in Philadelphia. We also need to specify a temporal quantization, which specifies how one discrete step maps to continuous time intervals. The temporal quantization is tuned programmatically to maximize the average entropy rate of the event streams, which results in 12 hour steps in case of the weather data, 3 days for the seismic prediction, and 1 or 2 days for urban crime. With the exception of the precipitation event in case of the weather modeling problem, the event frequencies are all lower than 10%. We also choose how far into future we make predictions (prediction horizon), which is chosen to be 1 week for weather, 4 months for seismic prediction, and $3-7$ days for urban crime (See Extended Data Tab. 1 for details).

Predictive ability of the FN framework is bench-marked against carefully tuned LSTM models[36]. The LSTM models use input dimension equal to the number of sequences, two fully connected hidden layers with LSTM units[37] ((No. units in the first layer, No. units in the second layer)=$(1000, 100)$ and $(2000, 500)$), and a time-distributed dense output layer[38], trained over 1000 epochs. We use tensorflow.keras package[39] for LSTM implementation with mean squared error as loss and adam as optimizer. Using cross-entropy (ce) as the loss function instead of mean square error did not produce significant difference in performance. We train LSTMs with the same data used for FNs, using binarized input sequences with 0 indicating absence of events and 1

the events of interest (See *Methods: LSTM Comparison* for details). We compare the achieved performance (See Tab. 1) of FN and LSTM architectures using 1) distribution of the area under the the receiver operating curve (AUC) for the spatial tiles in each application, 3) the precision-recall curves (PRC)

Performance Metrics

We use a flexible approach in evaluating AUC for both **`Fractal Net`** and LSTM; a positive prediction is treated as correct if there is at least one event recorded in ± 1 time steps in the target spatial tile. We also account for the spatial variability of the exact event location in the evaluation of PRC by replacing predicted events by 2D Gaussian densities followed by the choice of a decision threshold.

More specifically, for a fixed time step t, we construct a *risk map* by summing 2D Gaussian densities centered at tiles with a positive prediction. Finally, a threshold z is chosen so that tiles with a risk level above z is reported as having a positive prediction. The standard deviation and the z-levels are chosen in a manner a threshold is selected on a ROC curve.

FN significantly outperforms LSTM models in all metrics. Specifically, FN increases area under the receiver operating characteristic curve (AUC) by 10.0% for extreme weather, 13.5% for seismic activity over the local third quartile, and 17.7%, 10.0%, and 12.3% for criminal infractions in Atlanta, Chicago, and Philadelphia, respectively. FN boosts sensitivity at 90% positive predictive value by 161.9%, 191.3%, 150.0%, 418.6%, and 50.8% for the corresponding datasets (See Tab. 1). In addition to superior performance, our framework results in models with far fewer parameters (See Extended Data Tab. 2), likely due to the learning of stochastic generators rather than trying to encode every possible input variation[40], leading to a superior sample complexity.

Extended Data Tab. 1
Dataset Statistics, Data Sources, and Variable Explanation

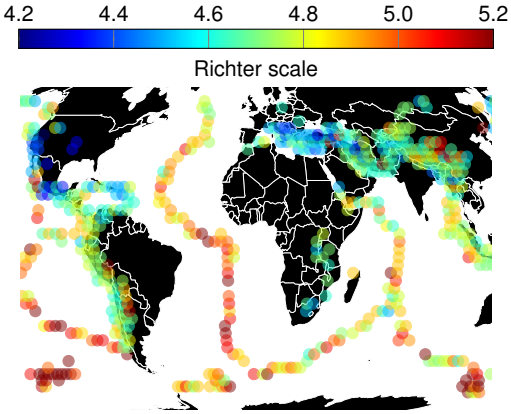
	US Weather	Earthquakes	Atlanta Crime	Chicago Crime	Philadelphia Crime
temporal res.	12 hrs	3 days	2 days	1 day	1 day
spatial res.	neighborhood of weather-station	$3^\circ \times 3^\circ$	$1006' \times 1006'$	$958' \times 1007'$	$907' \times 986'$
prediction horizon	7 days	120 days	6 days	7 days	3 days
train period	16/01/01 to 18/12/31	09/01/01 to 19/08/21	14/01/01 to 18/12/31	14/01/01 to 16/12/31	14/01/01 to 16/12/31
test period	19/01/01 to 19/04/30	19/08/21 to 20/08/21	19/01/01 to 19/04/10	17/01/01 to 17/04/10	17/01/01 to 17/04/10
source	https://smoosavi.org/datasets/lstw Original source: https://www.weather.gov/asos/	https://earthquake.usgs.gov/	https://data.world/brycantahb/crime-in-atlanta-2009-2017 Original source: http://opendata.atlantapd.org/	https://data.cityofchicago.org/browse?q=crime	https://www.opendataphilly.org/dataset/crime-incidents.
event description and frequency	*precipitation*: Fog, Storm, Rain (28.3%); *winter*: Snow, Cold (7.5%); *severe events*:10%.	Earthquakes of maginitude 4.0 and higher (2.46%).	*violent crime*: homicide, assault, battery (4.0%); *property crime*: burglary, theft, motor vehicle theft (4.7%);	*violent crime*: homicide, aggravated assault (7.6%); *property crime*: burglary, auto theft (6.9%).	*violent crime*: homicide, assault (8.1%); *property crime*: burglary, theft, motor vehicle theft (9.1%);

Extended Data Tab. 2
Prediction accuracy and model parameters comparison

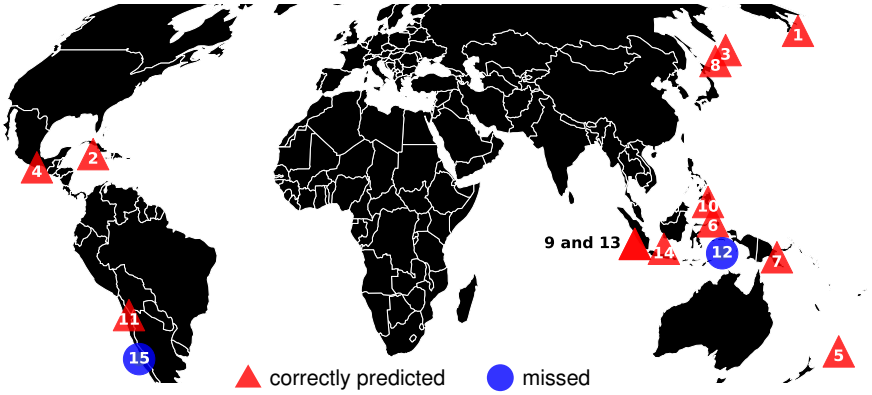
		US Weather	Earthquakes	Atlanta Crime	Chicago Crime	Philadelphia Crime
Fractal Net AUC	mean	.82112	.82241	.87845	.86237	.84091
	median	.81834	.82745	.88677	.85991	.84222
LSTM 1 AUC	mean	.73536	.72479	.74658	.78051	.74870
	median	.73951	.74983	.76324	.78294	.76118
LSTM 2 AUC	mean	.74667	.72003	.72628	.78373	.73780
	median	.75404	.75146	.74340	.78594	.75446
outperformance	mean	10.0%	13.5%	17.7%	10.0%	12.3%
	median	8.5%	10.4%	16.2%	9.4%	10.6%
Fractal Net No. parameters		4,800,618	436,551	106,253	3,211,478	440,005
Fractal Net avg. depth		30.1	4.78	4.17	15.3	8.49
LSTM 1 No. parameters		26,261,720	7,593,968	8,184,512	29,727,065	8,697,137
LSTM 2 No. parameters		66,235,320	27,538,768	28,762,912	73,418,655	29,825,537

1. We denote an **LSTM** by (no. units in the first layer, no. units in the second layer, epochs).
2. We consider two **LSTM**s: **LSTM** $1 = (1000, 100, 1000)$ and **LSTM** $2 = (2000, 500, 1000)$.
3. The **LSTM** model having the better performance for each dataset is highlighted.
4. The outperformance is calculated as (**Fractal Net** AUC − Better **LSTM** AUC)/(Better **LSTM** AUC).

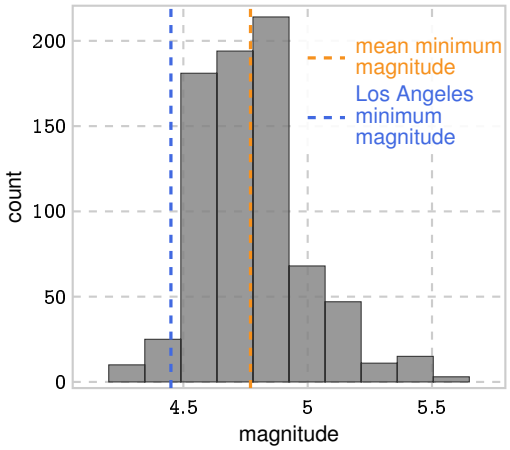
a. Minimum magnitudes used in defining target seismic events for prediction



c. Correct and missed predictions for 15 most significant events in period 22-08-2019 to 21-08-2020



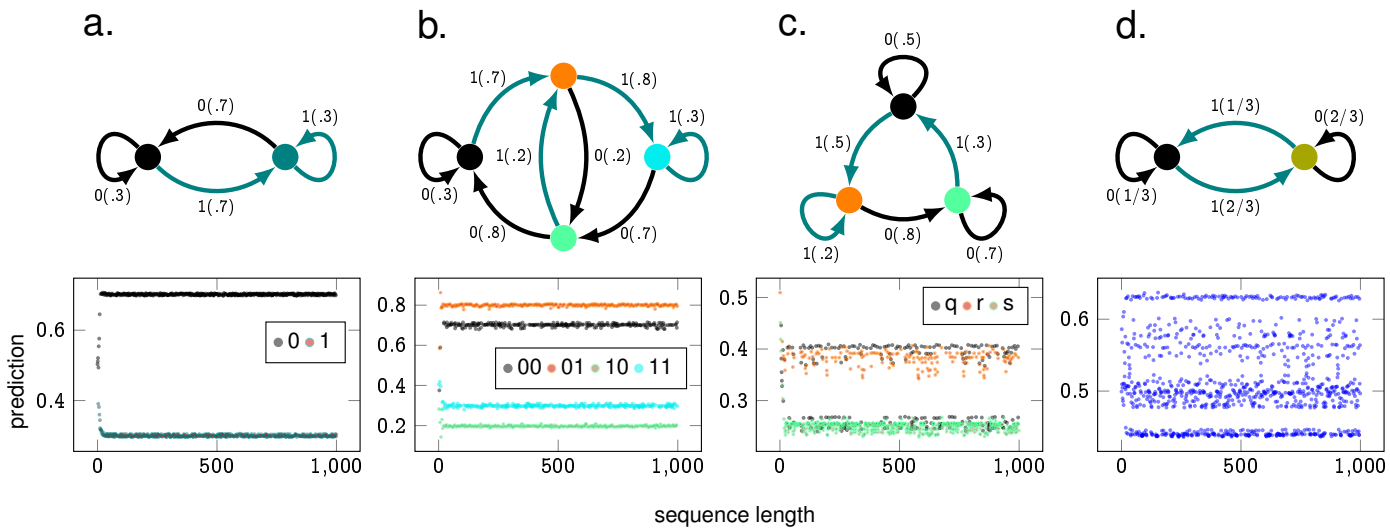
b. Distribution of minimum magnitudes of events targeted for prediction



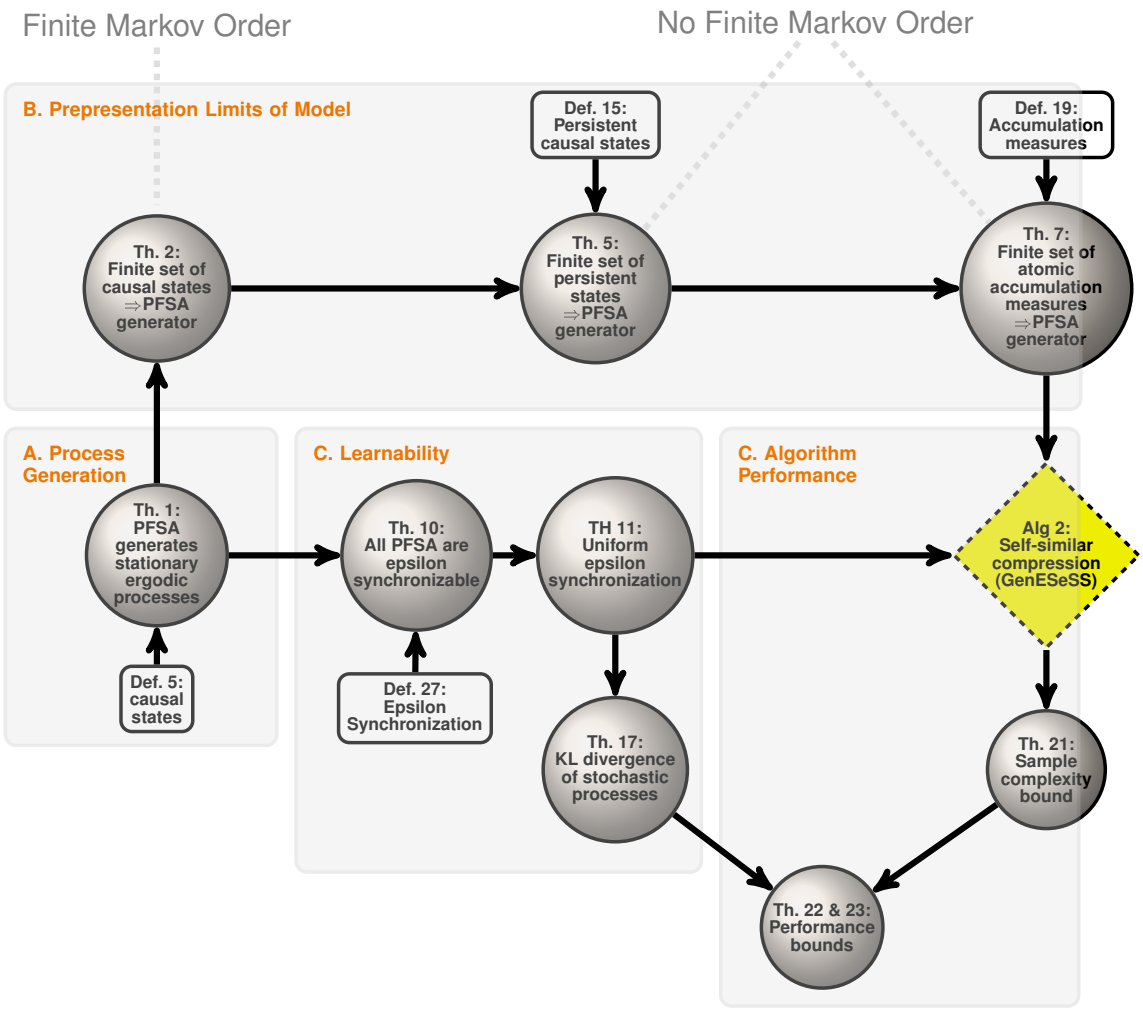
d. Top 15 events recorded between 22-08-2019 and 21-08-2020

	time	mag.	location
1	2020-07-22 06:12:44	7.8	105 km SSE of Perryville, Alaska
2	2020-01-28 19:10:24	7.7	123km NNW of Lucea, Jamaica
3	2020-03-25 02:49:21	7.5	221km SSE of Severo-Kuril'sk, Russia
4	2020-06-23 15:29:04	7.4	9 km SE of Santa María Xadani, Mexico
5	2020-06-18 12:49:53	7.4	south of the Kermadec Islands
6	2019-11-14 16:17:40	7.1	138km E of Bitung, Indonesia
7	2020-07-17 02:50:22	7.0	114 km NNW of Popondetta, Papua New Guinea
8	2020-02-13 10:33:44	7.0	94km ENE of Kuril'sk, Russia
9	2020-08-18 22:29:24	6.9	126 km WSW of Bengkulu, Indonesia
10	2019-12-15 06:11:51	6.8	7km S of Magsaysay, Philippines
11	2020-06-03 07:35:36	6.8	48 km SW of San Pedro de Atacama, Chile
12	2020-05-06 13:53:55	6.8	Banda Sea
13	2020-08-18 22:23:59	6.8	139 km WSW of Bengkulu, Indonesia
14	2020-07-06 22:54:47	6.7	98 km N of Batang, Indonesia
15	2019-09-29 15:57:53	6.7	69km WSW of Constitucion, Chile

Extended Data Fig. 1. **Minimum magnitudes used in defining target seismic events for prediction and performance on recent events of significance. Panel a** Minimum magnitude of events considered as prediction targets for locations around the globe. Note that the distribution of the minimum magnitudes in **panel b** illustrates a minimum magnitude of 4.2 and a maximum of 5.65 on the Richter scale with an average of 4.7. As a comparison, the minimum magnitude of target events near Los Angeles, USA is ≈ 4.5. Panel c and d shows recent high-magnitude events of maximum magnitude predicted correctly and missed in the out-of-sample prediction period between August 2019 and August 2020. As shown, we correctly predict 12 out of top 15 events (10 out of top 10), and do so 120 ± 3 days in advance.



Extended Data Fig. 2. **Neural nets need finite Markov order to succeed** We generate 100 sequences of length 5000 from each of the four stochastic processes generated by the models in the top row (See Examples of PFSA in Supplementary Text Sec. II). We train LSTM models[41,42] using the tensorFlow.keras package[39] with 50 hidden units, sigmoid function for activation, mean squared error (MSE) as loss function, 100 training epochs with batch size 32. Given our loss function, the LSTM model should learn to output the probability of producing 1 as the next output symbol. In the bottom row, we show the scatter plots of probability of 1-predictions of LSTM models vs input length. We color-code the dots in the scatter plots for the two processes of finite Markov order (Panel a and b), and the one generated by the three-state model (Panel c). If we know the current input leads to a certain state, we color the next output with the color of the state. The process in panel d has no finite Markov order and its generating model is not synchronizable. The predictions show that the LSTM fails to uncover the 1-probability distribution.



Extended Data Fig. 3. **Schematic Map of Theoretical Development.** The mathematical development detailed in the Supplementary text comes together as shown above to establish correctness of the inference algorithms, and performance and complexity bounds.

Algorithm 1: `Fractal Net`

Data:

- a set of sequence $\{x_i : \ i = 1, \ldots, N\}$ of length n;
- a hyperparameter $0 < \varepsilon < 1$;
- a model inference length $n_0 < n$;
- a maximal delay $\Delta_{\max}$;
- a threshold coefficient of causal dependence γ_0 for admissible models;

Result: A set of XPFSA models and a set of scalar weights for each target $r \in \{1, \ldots, N\}$.

/* **Infer models** */

Let $\mathcal{M}_r = \emptyset$ be the set of admissible models for each target $r \in \{1, \ldots, N\}$;
for *each delay* $\Delta = 1, \ldots, \Delta_{max}$ **do**
  for *each source* $s = 1, \ldots, N$ *and target* $r = 1, \ldots, N$ **do**
    Let $x_{\text{in}} = (x_s)_1^{n_0 - \Delta}$;
    Let $x_{\text{out}} = (x_r)_{\Delta+1}^{n_0}$;
    Calculate PFSA $G = \mathbf{GenESeSS}\,(x_{\text{in}}, \varepsilon)$;
    Calculate XPFSA $H^s_{r,\Delta} = \mathbf{xGenESeSS}\,(x_{\text{out}}, \varepsilon)$;
    Let $\gamma^s_{r,\Delta} = \mathbf{coefCausalDependence}(G, H^s_{r,\Delta})$;
    if $\gamma^s_{r,\Delta} \geq \gamma_0$ **then**
      Let $\mathcal{M}_r = \mathcal{M}_r \cup \left\{H^s_{r,\Delta}\right\}$;

/* **Learn scalar weights** */

for *each target* $r = 1, \ldots, N$ **do**
  Let $I_r = \left\{(s, \Delta) : \text{ there is a model } H^s_{r,\Delta} \in \mathcal{M}_r\right\}$;
  for *each timestamp* $t = 1, \ldots, n - n_0$ **do**
    Let $\mathbf{x}_t$ be a vector with index set I_r;
    for *each pair* $(s, \Delta) \in I_r$ **do**
      Let x_{in} the length l sub-sequence of x_s that ends in the $(n_0 + t - \Delta)$-th entry;
      Let the entry of $\mathbf{x}_t[s, \Delta] = \mathbf{predict}\left(H^s_{r,\Delta}, x_{\text{in}}\right)$;
      Let $y_t = x_r[n_0 + t]$;
  Let X the matrix with the t-th row being $\mathbf{x}_t$;
  Let $\mathbf{y}$ be the vector with the t-th entry being y_t;
  Initialize a suitable regressor **Reg**;
  Get scalar weights $\mathbf{w}_r = \left(w^s_{r,\Delta}\right)_{(s,\Delta)\in I_r} = \mathbf{Reg}\,(X, \mathbf{y})$;
return $\{(\mathcal{M}_r, \mathbf{w}_r) : \ r = 1, \ldots, N\}$;

Algorithm 2: GenESeSS

Data: A sequence x over alphabet Σ, $0 < \varepsilon < 1$
Result: State set Q, transition map δ, and transition probability $\tilde{\pi}$
/* Step One: Approximate ε-synchronizing sequence */
Let $L = \left\lceil \log_{|\Sigma|} 1/\varepsilon \right\rceil$;
Calculate the **derivative heap** $\mathcal{D}_\varepsilon^x$ equaling $\left\{ \hat{\phi}_y^x \ : \ y \text{ is a sub-sequence of } x \text{ with } |y| \leq L \right\}$;
Let $\mathcal{C}$ be the convex hull of D_ε^x;
Select x_0 with $\hat{\phi}_{x_0}^x$ being a vertex of $\mathcal{C}$ and has the highest frequency in x;
/* Step Two: Identify transition structure */
Initialize $Q = \{q_0\}$;
Associate to q_0 the **sequence identifier** $x_{q_0}^{\mathrm{id}} = x_0$ and the probability vector $d_{q_0} = \hat{\phi}_{x_0}^x$;
Let $\tilde{Q}$ be the set of states that are just added and initialize it to be Q;
while $\tilde{Q} \neq \emptyset$ **do**
    Let $Q_{\mathrm{new}} = \emptyset$ be the set of new states;
    for $(q, \sigma) \in \tilde{Q} \times \Sigma$ **do**
        Let $x = x_q^{\mathrm{id}}$ and $d = \hat{\phi}_{x\sigma}^x$;
        if $\|d - d_{q'}\|_\infty < \varepsilon$ *for some* $q' \in Q$ **then**
            Let $\delta(q, \sigma) = q'$;
        else
            Let $Q_{\mathrm{new}} = Q_{\mathrm{new}} \cup \{q_{\mathrm{new}}\}$ and $Q = Q \cup \{q_{\mathrm{new}}\}$;
            Associate to q_{new} the sequence identifier $x_{q_{\mathrm{new}}}^{\mathrm{id}} = x\sigma$ and the probability vector $d_{q_{\mathrm{new}}} = d$;
            Let $\delta(q, \sigma) = q_{\mathrm{new}}$;
    Let $\tilde{Q} = Q_{\mathrm{new}}$;
Take a strongly connected subgraph of the labeled directed graph defined by Q and δ, and denote the vertex set of the subgraph again by Q;
/* Step Three: Identify transition probability */
Initialize counter $N[q, \sigma]$ for each pair $(q, \sigma) \in Q \times \Sigma$;
Choose a random starting state $q \in Q$;
for $\sigma \in x$ **do**
    Let $N[q, \sigma] = N[q, \sigma] + 1$;
    Let $q = \delta(q, \sigma)$;
Let $\tilde{\pi}(q) = \left[\!\left[(N[q, \sigma])_{\sigma \in \Sigma} \right]\!\right]$;
return Q, δ, $\tilde{\pi}$;

Algorithm 3: `xGenESeSS`

Data: A sequence x_{in} over alphabet Σ_{in}, a sequence x_{out} over alphabet Σ_{out}, and $0 < \varepsilon < 1$
Result: State set R, transition map η, and output probability χ

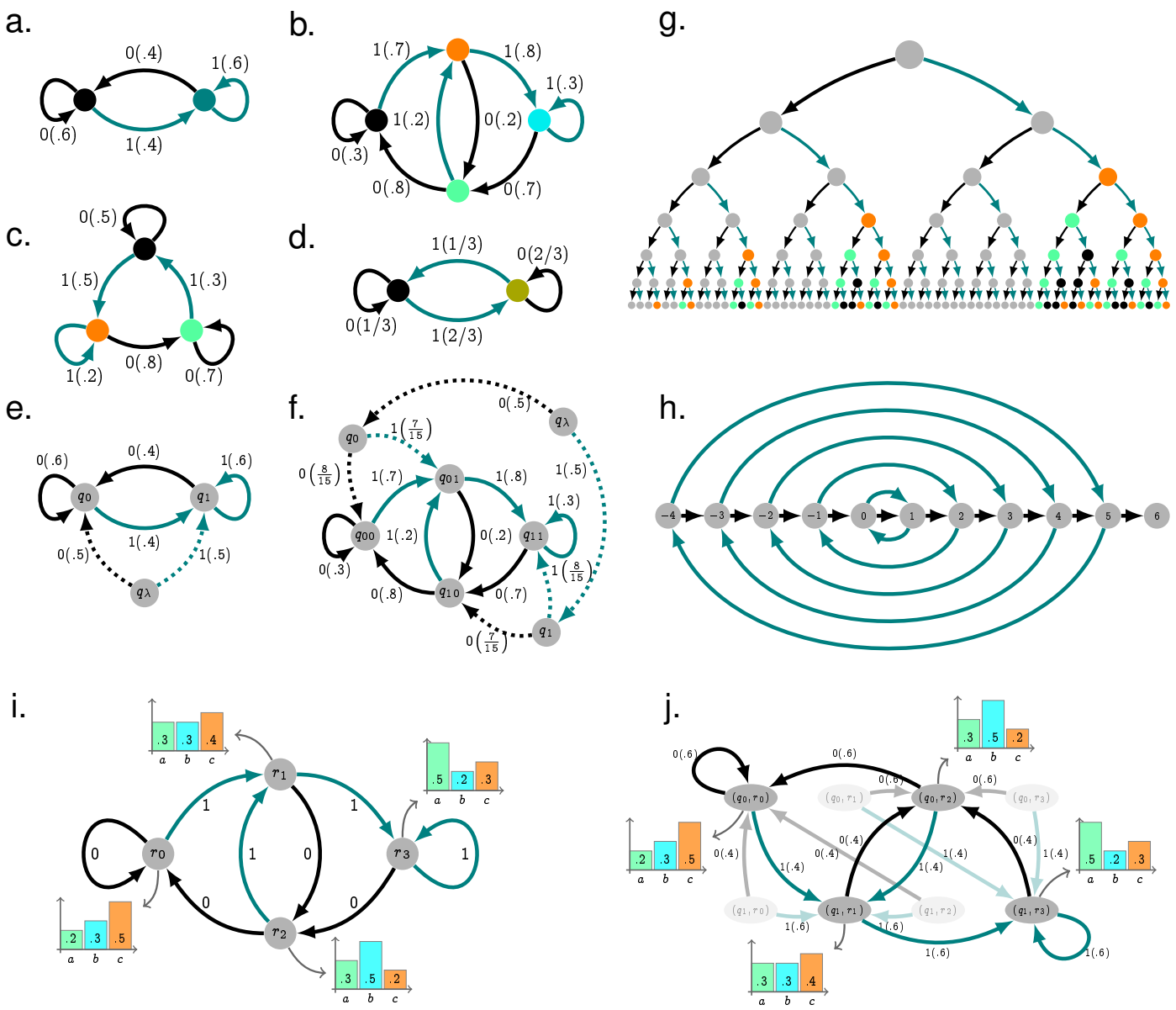
/* **Step One: Approximate ε-synchronizing sequence** */
Let $L = \left\lceil \log_{|\Sigma_{\text{in}}|} 1/\varepsilon \right\rceil$;
Calculate **cross derivative heap** $\mathcal{D}_\varepsilon^{x_{\text{in}}, x_{\text{out}}}$ equaling $\left\{ \hat{\phi}_y^{x_{\text{in}}, x_{\text{out}}} \ : \ y \text{ is a sub-sequence of } x_{\text{in}} \text{ with } |y| \leq L \right\}$;
Let $\mathcal{C}$ be the convex hull $\mathcal{D}_\varepsilon^{x_{\text{in}}, x_{\text{out}}}$;
Select x_0 with $\hat{\phi}_{x_0}^{x_{\text{in}}, x_{\text{out}}}$ being a vertex of $\mathcal{C}$ and has the highest frequency in x;
/* **Step Two: Identify transition structure** */
Initialize $R = \{r_0\}$;
Associate to r_0 the **sequence identifier** $x_{r_0}^{\text{id}} = x_0$ and the probability vector $\chi(r_0) = \hat{\phi}_{x_0}^{x_{\text{in}}, x_{\text{out}}}$;
Let $\tilde{R}$ be the set of states that are just added and initialize it to be R;
while $\tilde{R} \neq \emptyset$ **do**
    Let $R_{\text{new}} = \emptyset$ be the set of new states;
    for $(r, \sigma) \in \tilde{R} \times \Sigma_{in}$ **do**
        Let $x = x_r^{\text{id}}$ and $d = \hat{\phi}_{x\sigma}^{x_{\text{in}}, x_{\text{out}}}$;
        if $\|d - \chi(r')\|_\infty < \varepsilon$ *for some* $r' \in R$ **then**
            Let $\eta(r, \sigma) = r'$;
        else
            Let $R_{\text{new}} = R_{\text{new}} \cup \{r_{\text{new}}\}$ and $R = R \cup \{r_{\text{new}}\}$;
            Associate to r_{new} the sequence identifier $x_{r_{\text{new}}}^{\text{id}} = x\sigma$ and the probability vector $\chi(r_{\text{new}}) = d$;
            Let $\eta(r, \sigma) = r_{\text{new}}$;
    Let $\tilde{R} = R_{\text{new}}$;
Take a strongly connected subgraph of the labeled directed graph defined by R and η, and denote the vertex set of the subgraph again by R;
/* **Step Three: Identify output probability** */
Initialize counter $N[r, \tau]$ for each pair $(r, \tau) \in R \times \Sigma_{\text{out}}$;
Choose a random starting state $r \in R$;
for $i \in 1, \ldots, |x_{in}|$ **do**
    Let σ_i be the i-th symbol in x_{in} and τ_i be the i-th symbol in x_{out};
    Let $N[r, \tau_i] = N[r, \tau_i] + 1$;
    Let $r = \eta(r, \sigma_i)$;
Let $\chi(r) = \left[\!\left[(N[r, \tau])_{\tau \in \Sigma_{\text{out}}} \right]\!\right]$;
return R, η, χ;

Extended Data Tab. 3
Causality table of the process in Example 1

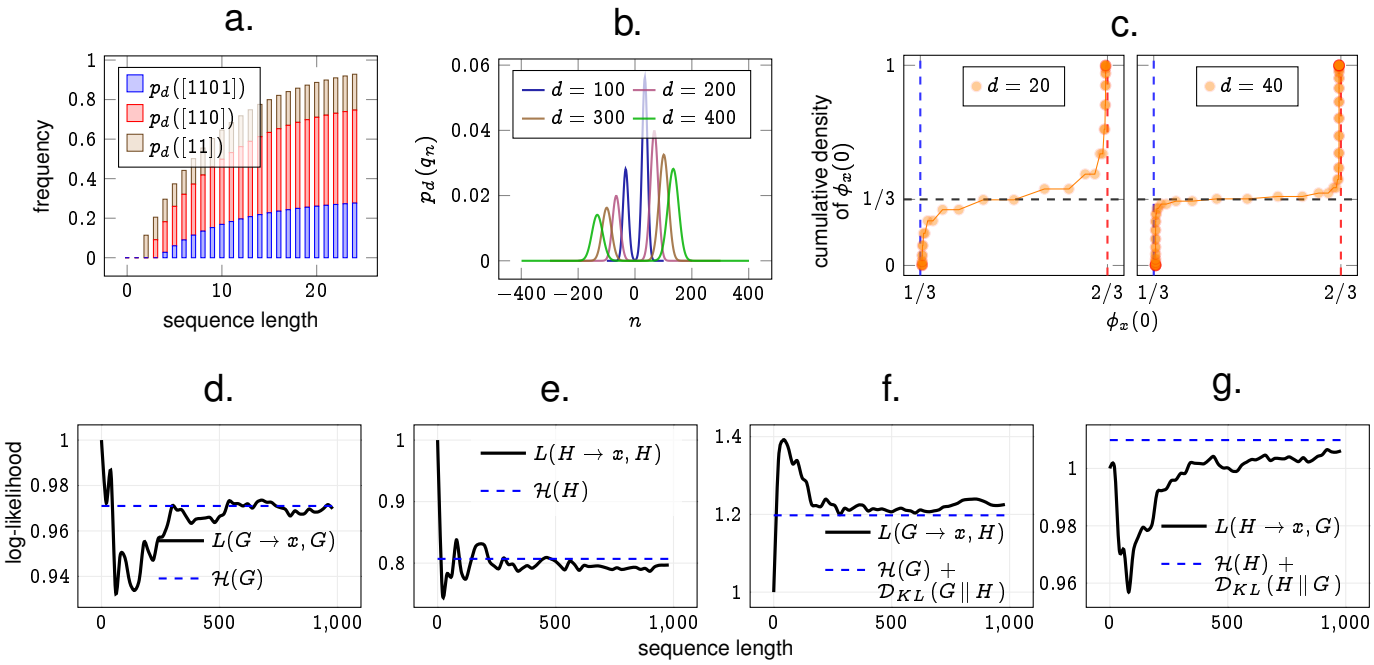
$x \in \Sigma^\star$	$\mu(x\Sigma^\omega)$	ϕ_x	causal state
λ	1	$(.5, .5)$	q_λ
0	.5	$(.6, .4)$	q_0
1	.5	$(.4, .6)$	q_1
00	.3	$(.6, .4)$	q_0
01	.2	$(.4, .6)$	q_1
10	.2	$(.6, .4)$	q_0
11	.3	$(.4, .6)$	q_1
000	.18	$(.6, .4)$	q_0
001	.12	$(.4, .6)$	q_1
010	.08	$(.6, .4)$	q_0
011	.12	$(.4, .6)$	q_1
100	.12	$(.6, .4)$	q_0
101	.08	$(.4, .6)$	q_1
110	.12	$(.6, .4)$	q_0
111	.18	$(.4, .6)$	q_1
⋮	⋮	⋮	⋮

Extended Data Tab. 4
Causality table of the process in Example 3

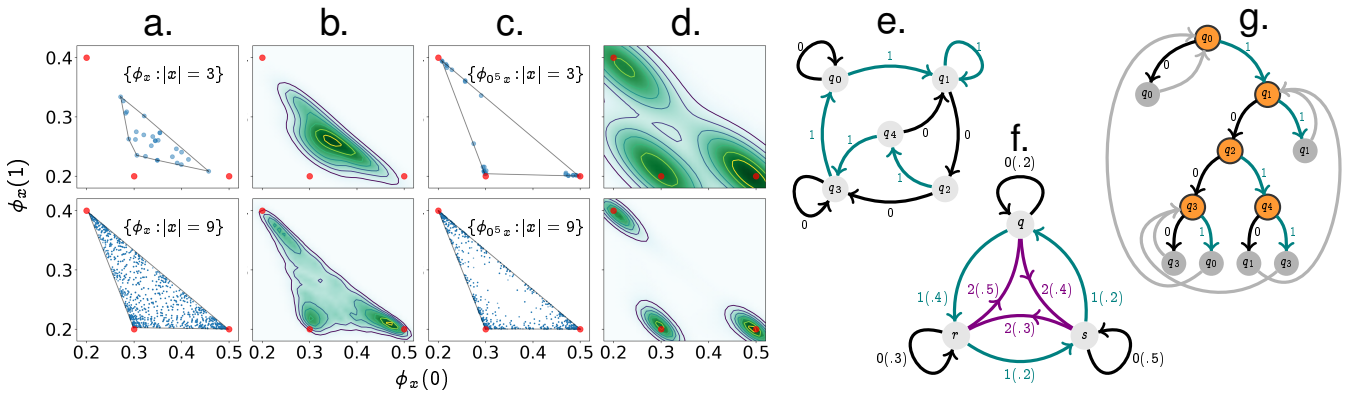
$x \in \Sigma^\star$	$\mu(x\Sigma^\omega)$	ϕ_x	causal state
λ	1	$(1/2, 1/2)$	q_λ
0	1/2	$(8/15, 7/15)$	q_0
1	1/2	$(7/15, 8/15)$	q_1
00	4/15	$(3/10, 7/10)$	q_{00}
01	7/30	$(1/5, 4/5)$	q_{01}
10	7/30	$(4/5, 1/5)$	q_{10}
11	4/15	$(7/10, 3/10)$	q_{11}
000	2/25	$(3/10, 7/10)$	q_{00}
001	14/75	$(1/5, 4/5)$	q_{01}
010	7/150	$(4/5, 1/5)$	q_{10}
011	14/75	$(7/10, 3/10)$	q_{11}
100	14/75	$3/10, 7/10$	q_{00}
101	7/150	$(1/5, 4/5)$	q_{01}
110	14/75	$(4/5, 1/5)$	q_{10}
111	2/25	$(7/10, 3/10)$	q_{11}
⋮	⋮	⋮	⋮



Extended Data Fig. 4. **Examples of PFSA in Supplementary Text Sec. II and in Supplementary Text Sec. VII.** Panel a: an $M2$ PFSA. The PFSA generates a Markov process, which gives rise to the labeled directed graph in Panel e by subtree stitching. We note that the graph is not strongly connected since the causal state q_λ has no in-coming edge. The unique strongly connected component of the graph, *i.e.*, the induced subgraph on q_0 and q_1, is exactly the generating $M2$ PFSA. See Supplementary Text Example 1 for detail. Panel b: an $M4$ PFSA. The PFSA generates a process of Markov order 2, which gives rise to the labeled directed graph in Panel f by subtree stitching. We note that the graph is not strong connected, since the causal states q_λ, q_0, and q_1 have no in-coming edge. The unique strongly connected component, *i.e.*, the induced subgraph on q_{ij}, $i, j \in \{0, 1\}$. is exactly the generating $M4$ PFSA. See Supplementary Text Example 3 for detail. Panel c: a synchronizable 3-state PFSA discussed in Supplementary Text Example 4. The shortest synchronizing sequence of the PFSA is 11. Part of the binary tree given rise by the process generated by the PFSA is demonstrated in Panel g. The nodes in the tree are sequences, with the root being the empty sequence. The darker nodes in the tree represent sequences containing 11 and hence are synchronizing. There are higher fraction of synchronizing nodes with longer sequences. Panel d: a non-synchronizable S PFSA discussed in Supplementary Text Example 5. The labeled directed graph obtained by subtree stitching is an infinite graph, and we demonstrate part of the graph in Panel f. Thgge node labeled by n represents the causal state $[x]$ with $\phi_x \dot{\sim} \left(.5^{n+1} + 1, .5^n + .5\right)$ in Supplementary Text Eq. (41). Panel i: Example of an XPFSA with four states. The bar charts show the output probability vectors $\chi(r_i)$, $i = 0, 1, 2, 3$. For example, at state r_0, the probability of getting symbol a, b, and c as output is .2, .3, and .5, respectively. We note that an XPFSA *doesn't* have a transition probability map as a PFSA does. Panel j: the synchronous composition of the $M2$ PFSA in Fig. 4a to the XPFSA in Panel a. We show the composition over $\{q_0, q_1\} \times \{r_0, r_1, r_2, r_3\}$ but highlight only the strongly connected component that is kept for the synchronous composition.



Extended Data Fig. 5. **Examples in Supplementary Text Sec. III and example 11 in Supplementary Text Sec. VI** Panel a: Sum of probabilities of causal states $[11]$, $[110]$, and $[1101]$ for sequence length $d = 0, 1, \ldots, 25$ as discussed in Example 6. The sum of probabilities of sequences synchronzing to the three causal states approaches 1 as d increases. Hence, although the process has infinitely many causal states, it is generated by a PFSA. Panel b: $p_d(q_n)$ vs n for sequence length $d = 100, 200, 300, 400$ as discussed in Example 7. The curves keep on flattening with increasing d, which implies that no causal state q_n is persistent. The cumulative probability density functions of $\phi_x(0)$ with sequence lengths $d = 20, 40$ is demonstrated in Panel c for this process. For each fixed d, the abscissa is a value $h \in \{\phi_x(0) : x \in \Sigma^d\}$, and the ordinate is $p_d\{x \in \Sigma^d : \phi_x(0) \leq h\}$. The curves approaches a step function as d increases, which implies the set of causal states has two cumulation points μ_1 and μ_2 with $\mu_1^1 = (1/3, 2/3)$ and $\mu_2^1 = (2/3, 1/3)$. The two accumulation points correspond to q_∞ and $q_{-\infty}$ as in Eq. (60). Panel d-g: examples of log-likelihood convergence to the sum of entropy rate and KL divergence. The horizontal line in each plot shows the limit the log-likelihood should approach to as length of sequence increases.



Extended Data Fig. 6. **Examples in Supplementary Text Sec. VIII** For the non-synchronizable 3-state 3-symbol PFSA G in Panel f, and for sequence length $d = 3$ and 9, we show in Panel a the scatter plots of $\{\phi_x(0, 1) : |x| = d\}$, in Panel b the density plots of the previous set with the weight of the point $\phi_x(0, 1)$ being $p_G(x)$, in Panel c the scatter plots of $\{\phi_{0^5x}(0, 1) : |x| = d\}$, and in Panel d the density plots of the previous set with the weight of the point $\phi_{0^5x}(0, 1)$ being $p_G(x|0^5)$. Panel e is a PFSA with 5 states and over binary alphabet, and Panel g is a running tree of **GenESeSS** in case x_0 find in Extended Data Algorithm 2 line 4 is a synchronizing sequence to state q_0. A new state is colored orange, and a repeated state is colored gray.