Lurie Networks with Robust Convergent Dynamics

Carl Richardson[1,2], Matthew Turner[1], Steve Gunn[1]

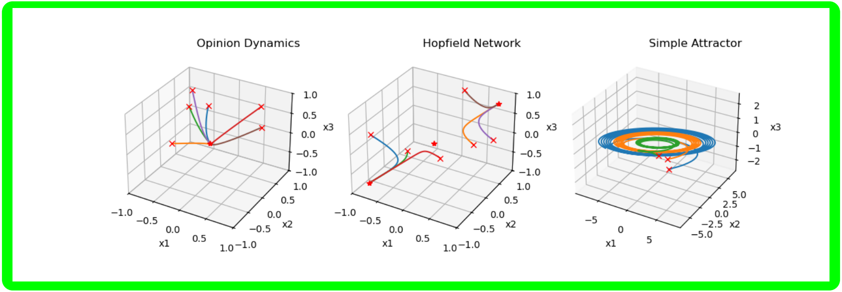
University of Southampton[1]
The Alan Turing Institute[2]

The Alan Turing Institute

Motivation I

Many dynamical systems exhibit some form of convergence. These include convergence to unique or non-unique equilibrium points and limit cycles. The examples below (from left to right) illustrate:

- how three agents converge to the same opinion
- the evolution of three neurons in a Hopfield network
- autocatalytic chemical reactions (e.g., of a Rössler system)



Motivation II

Of course, the brain is also a dynamical system formed of interacting neural circuits (*graph coupled networks*) implementing a myriad of functions, some of which leverage convergent (attractor) dynamics[1] for:

- learning
- memory storage
- robustness
- information processing

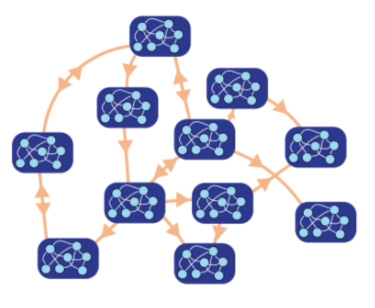


Figure: L. Kozachkov, et al., "RNNs of RNNs: Recursive construction of stable assemblies of recurrent neural networks.", NeurIPS, 2022.

[1] Mikail Khona and Ila R Fiete. Attractor and integrator networks in the brain. Nature Reviews Neuroscience, 23(12):744–766, 2022.

Research Questions

Technical

1. Can we encode convergence into a single artificial network?
2. How can we assemble a graph coupled network which also encodes convergence?

Practical

1. Does this result in more accurate and robust models of dynamical systems?
   - Potentially useful modelling tool for neuroscience!
2. Can this be used to construct a general and robust ML model?
   - E.g., multimodal applications

Outline

1. Propose the *Lurie network* as a unifying recurrent model
2. Derive constraints on its weights to ensure convergence
3. Parametrise the weights so constraints are always satisfied
4. Construct a graph coupled network
5. Derive constraints and a parametrisation of the graph connections to ensure the graph coupled network is convergent

Lurie Network I

$$\begin{aligned}
\dot{x}(t) &= \overbrace{Ax(t) + B\Phi(y(t)) + b_x}^{:=f(x(t))} & \qquad x \in \mathbb{R}^n \\
y(t) &= Cx(t) + b_y & \qquad y \in \mathbb{R}^n
\end{aligned}$$

- Defined as a continuous ODE with weights A, B, C and biases b_x, b_y. The function $\Phi(\cdot)$ represents the activation functions applied element-wise.
- Linear state space models, graph coupled oscillators and various RNNs can be viewed as special cases.
  - E.g., when $A = -I$, $C = I$ and $b_y = 0$ Lurie network $\rightarrow$ Wilson-Cowan RNN

Lurie Network II

$$
\begin{aligned}
x_{t+1} &= x_t + \int_t^{t+\Delta t} f(x(t)) \cdot dt \\
&\approx x_t + \Delta t \cdot \text{scheme}[f, x_t, \Delta t]
\end{aligned}
$$

- To train the model, the continuous ODE is discretised by recursively applying one step of a numerical integrator.
- For a regression task, x_0 is the input and the sampled trajectory $\{x_1, x_2, \ldots, x_N\}$ is the target.
- For a classification task, x_0 is the input and the classification is made according to a decoder function $g(x_N)$.

Convergence Constraint

Theorem 1

Consider the Lurie network with $\Phi(y) := \begin{bmatrix} \phi_1(y_1) & \dots & \phi_m(y_m) \end{bmatrix}^\top$ being slope-restricted such that $0 \leq \phi_j'(y_j) \leq g$. Fix $k \in \{1, 2, 3\}$ and define $\alpha_k := (2k)^{-1} \sum_{i=1}^{k} \lambda_i(A + A^\top)$. If $\alpha_k < 0$ and

$$z_k := g^2 \sum_{i=1}^{k} \sigma_i^2(B) \sigma_i^2(C) < \alpha_k^2 k$$

then the Lurie network has convergent dynamics.

- $\lambda_i(Z), \sigma_i(Z)$ denote the i^{th} largest eigen/singular value of Z.
- Choosing $k = 1, 2, 3$ respectively corresponds to convergence to unique equilibrium, multiple equilibrium or limit cycles.
- Enforcing convergence during training is naturally a constrained optimisation ...

Lurie Network with Convergent Dynamics

Theorem 2

Given $g > 0$, $k \in \{1,2,3\}$, $U_A, U_B, V_B, U_C, V_C \in \mathcal{O}(n)$, $G_A, \Sigma_B, \Sigma_C \in \mathcal{D}_+^n$, $Y_A \in \mathrm{Skew}(n)$ *and define*

$$A := \frac{1}{2} U_A \Sigma_A U_A^\top + \frac{1}{2} Y_A \qquad \Sigma_A := -\sqrt{\frac{4z_k}{k}} I_n - G_A$$

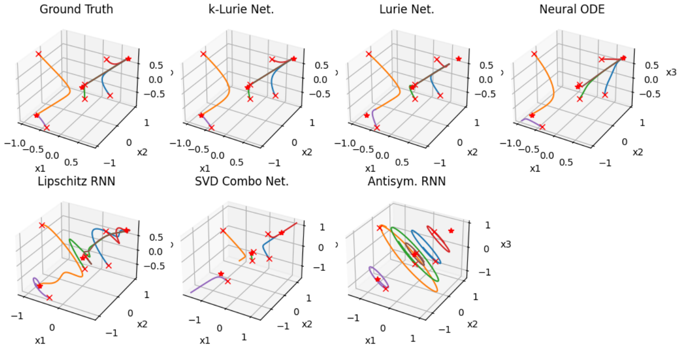
$$B := U_B \Sigma_B V_B^\top \qquad C := U_C \Sigma_C V_C^\top$$

then Theorem 1 is always satisfied.

- $\mathcal{O}(n), \mathrm{Skew}(n), \mathcal{D}_+^n$ respectively denote the sets of orthogonal, skew-symmetric and positive diagonal matrices of dimension n.
- g, k are hyperparameters. The orthogonal, skew-symmetric and positive diagonal matrices can all be implemented as parametrised layers[2] - these are the model parameters.
- Enforcing convergence during training is now unconstrained!

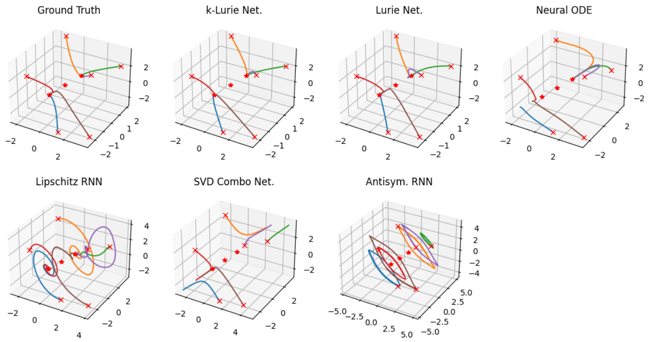
[2] https://docs.pytorch.org/tutorials/intermediate/parametrizations.html

Example: Hopfield Network



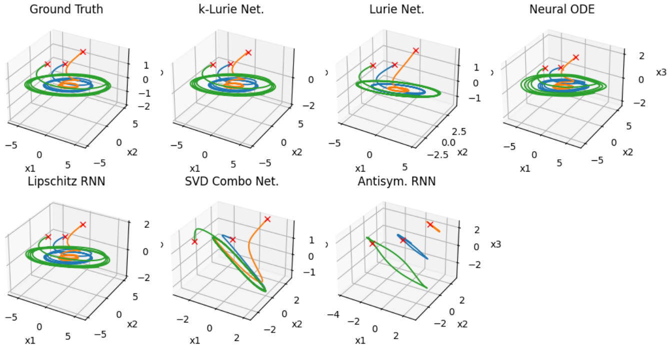
- Lurie Net. does not have convergent dynamics constraint.
- k-Lurie Net. has convergent dynamics constraint with $k = 2$ (Theorem 2).

Example: Hopfield Network OOD



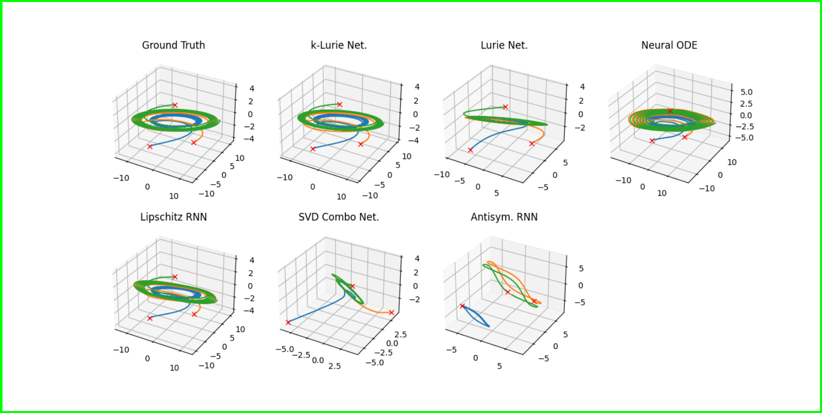
- Initial conditions (model inputs) are outside the training distribution (OOD).

Example: Simple Attractor



- Lurie Net. does not have convergent dynamics constraint.
- k-Lurie Net. has convergent dynamics constraint with $k = 3$ (Theorem 2).

Example: Simple Attractor OOD



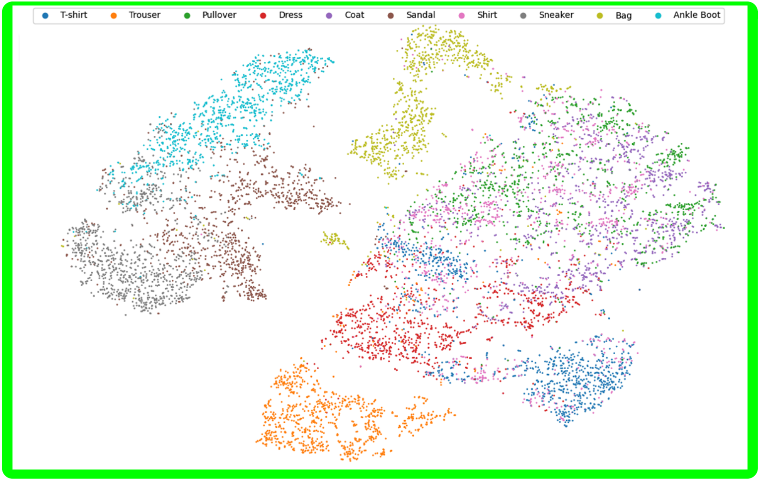
- Initial conditions (model inputs) are outside the training distribution (OOD).

Example: F-MNIST

Model	# Params.	Acc. (%)
LR-Net	$1,028,234$	95.03
Inception v3	$23,851,784$	94.44
CNN + Wilson-Cowan RNN	$5,179,521$	91.35
Wilson-Cowan RNN	-	88.39
CNN + k-Lurie Network	$1,057,994$	91.33
CNN + Lurie Network	$1,057,994$	90.95
Lurie Network	$1,853,386$	89.64
k-Lurie Network	$1,853,386$	85.78

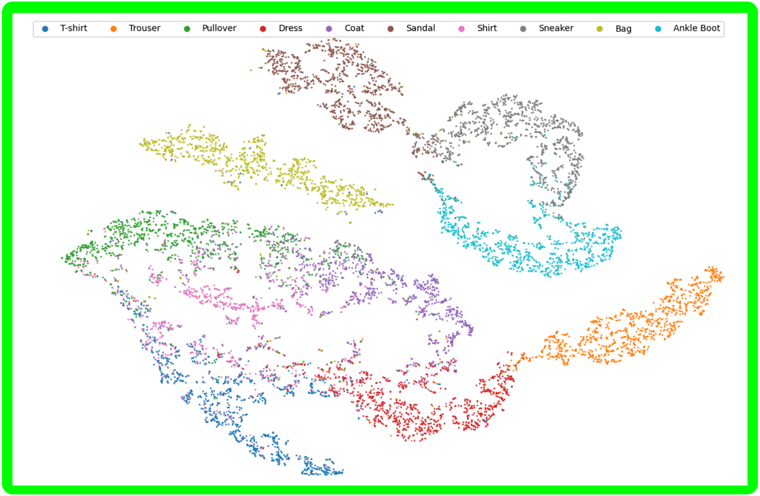
- Lurie Net. does not have convergent dynamics constraint.
- k-Lurie Net. has convergent dynamics constraint with $k = 3$ (Theorem 2).

Example: F-MNIST



- tSNE projection of Lurie network without convergence constraint.

Example: F-MNIST



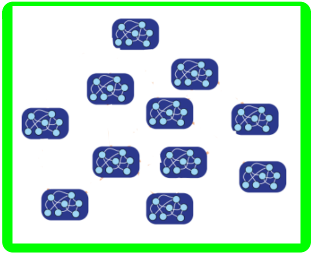
- tSNE projection of Lurie network with convergence constraint (Theorem 2).

Collection of Independent Lurie Networks

$$\dot{x}_q(t) = A_q x_q(t) + B_q \Phi(y_q(t)) + b_{x,q}$$
$$y_q(t) = C_q x_q(t) + b_{y,q}$$

$$\Updownarrow$$

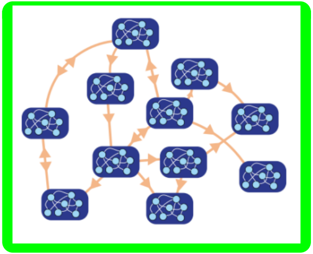
$$\dot{x}(t) = Ax(t) + B\Phi(y(t)) + b_x$$
$$y(t) = Cx(t) + b_y$$



- $q \in \{1, 2, \ldots, Q\}$ where Q denotes the # of individual Lurie networks to be connected together.
- $x(t) = \begin{bmatrix} x_1^\top(t) & \ldots & x_Q^\top(t) \end{bmatrix}^\top$, $y(t) = \begin{bmatrix} y_1^\top(t) & \ldots & y_Q^\top(t) \end{bmatrix}^\top$
- $b_x = \begin{bmatrix} b_{x,1}^\top & \ldots & b_{x,Q}^\top \end{bmatrix}^\top$, $b_y = \begin{bmatrix} b_{y,1}^\top & \ldots & b_{y,Q}^\top \end{bmatrix}^\top$
- $A = \mathrm{blockdiag}(A_1, \ldots, A_Q)$, $B = \mathrm{blockdiag}(B_1, \ldots, B_Q)$, $C = \mathrm{blockdiag}(C_1, \ldots, C_Q)$

Graph Lurie Network (GLN)

$$\dot{x}(t) = Ax(t) + B\Phi(y(t)) + b_x + Lx(t)$$
$$y(t) = Cx(t) + b_y$$



- A, B, C, b_x, b_y defined on previous slide.
- L is a block matrix $L = \left[L_{pq}\right] \in \mathbb{R}^{Qn\times Qn}$ where the block L_{pq} represents the connections from Lurie network q to Lurie network p.
- Essentially a Lurie network like we have already seen, but with sparse parameters encoding the graph structure.

Graph Convergence Constraint

Theorem 3

Fix $k \in \{1, 2, 3\}$. Consider a GLN where the Q independent Lurie networks satisfy Theorem 1. If the graph coupling matrix $L \in \mathbb{R}^{Qn \times Qn}$ satisfies

$$P^{(k)} L^{[k]} + (L^{[k]})^\top P^{(k)} \preceq 0$$

for $P \in \mathcal{D}_{+}^{Qn}$, then the GLN also has convergent dynamics.

- $Z^{(k)}, Z^{[k]}$ denote the k-multiplicative and k-additive compound matrices of Z. Conceptually not important - just algebra!

Graph Lurie Network with Convergent Dynamics

Theorem 4

Given $k \in \{1, 2, 3\}$*,* $G_L \in \mathbb{R}^{Qn \times Qn}$*,* $\Theta = \mathrm{blockdiag}(\Theta_1, \ldots, \Theta_Q)$ *where* $\Theta_q \in \mathcal{S}_+^n$ *for* $q \in \{1, \ldots, Q\}$*,* $P = \Theta^\top \Theta$ *and define*

$$L := G_L - P^{-1} G_L^\top P$$

then Theorem 3 is always satisfied.

- $\mathcal{S}_+^n$ denotes the set of positive definite matrices with dimension n.
- Using Theorem 2 and Theorem 4 enforces convergence during training of the GLN.
- Could use Theorem 2 to randomly generate fixed convergent Lurie networks and just learn connections of GLN. This has connections to *facilitated variation*[3] [4].

---

[3] J. Gerhart, et al., “The theory of facilitated variation,” 2007.

[4] L. Kozachkov, et al., ”RNNs of RNNs: Recursive construction of stable assemblies of RNNs,” 2022

Example: Graph Coupled Networks

Model	MSE (mean ± std, best) GC Hopfield	GC Attractor
GLN	0.016 ± 0.0003, **0.016**	0.293 ± 0.1969, **0.015**
k-Lurie Network	0.238 ± 0.0011, 0.237	0.737 ± 0.0108, 0.723
Lurie Network	2.537 ± 1.3327, 1.157	291.8 ± 191.84, 21.83
Neural ODE	0.138 ± 0.0291, 0.114	3.445 ± 0.3626, 2.942
Lipschitz RNN	0.124 ± 0.0161, 0.105	0.658 ± 0.1604, 0.433
SVD Combo	0.339 ± 0.1363, 0.229	3.024 ± 1.1240, 1.435
Antisym. RNN	0.448 ± 0.0023, 0.444	6.386 ± 0.0066, 6.380

- Lurie Net. does not have convergent dynamics constraint.
- k-Lurie Net. has convergent dynamics constraint with $k = 2$ for GC Hopfield and $k = 3$ for GC Attractor (Theorem 2).
- GLN has graph structured has convergent dynamics constraint with $k = 2$ for GC Hopfield and $k = 3$ for GC Attractor (Theorem 2, Theorem 4).

Conclusion

Technical

- Encoded convergence into a single artificial network.
- Assembled a graph coupled network which also encoded convergence.

Practical

- Showed multiple examples where prediction/classification accuracy and robustness improved.
- Future work: Can this be used to construct a general and robust ML model for multimodal applications?