Functional Measurement of Dynamical Robustness in Networks of RNNs

Joel Ye

Computer Science

Georgia Institute of Technology

joel.ye@gatech.edu

Abstract:
Network robustness is typically studied in models where nodes fail in discrete ways at simple thresholds. In recurrent computational networks, such as graphs of RNNs, this abstraction may be insufficient. We investigate how noise affects information processing in a recurrent graph neural network, as measured by a reduction in end-task performance. Specifically, we characterize how performance degrades with respect to noise strength and injection location, relating results with the underlying network structure. We also describe how network regularization improves functional robustness. Code: github.com/joel99/noised-rnn-networks.

Keywords: Network Robustness

1 Introduction

Computation in complex systems often involves the interaction of many components with individual dynamics. What happens when the activity of a part of the network becomes irregular? How well a system functions in such a scenario is termed "dynamical robustness." In network science, dynamical robustness stands in contrast to the more commonly studied structural robustness, which evaluates the robustness in the context of component failure. However, the study of robust network dynamics is at least as important as the study of the underlying network structure, and its failure modes can be qualitatively entirely different [1]. Real-world networks, especially biological or computational networks [1], often have nodes with internal dynamics. For example, some forms of epileptic seizure in the brain results from hyper-synchronization of neural circuits [2]. Such real-world complex node dynamics limit the applicability of analyses made with simplifying assumptions, like critical failure thresholds in cascading failure.

Thus, one core unstudied question in dynamical robustness is evaluating whether the binary failure of nodes (*i.e.* in structural analyses) is a suitable abstraction for nodes with complex internal dynamics. How do failure cascades translate to a dynamical context? To answer this, we can study noise propagation, which we define as how noising the state of one node affects the overall network. Understanding noise propagation could lead to better measures for preventing or intervening in dynamical network failures, with potential applications in the medical (epilepsy) or deep learning (making models robust to attack) fields.

In the real world, it is difficult to measure the dynamical state of individual nodes (e.g. in high quality brain scans), so we instead study the topic in simulation. Simulating data has the added benefit of providing white box control over network activity. [1][3][4][5] study robustness of simple oscillator activity, or systems with dynamics governed by simple state equations, to enable some theoretical analysis. However, abstracting dynamical state into an "activity" scalar compresses the richness of the actual dynamical activity, and we argue, again limits the practicality of studying dynamical state. Differently, we emphasize robustness of function, and study the preservation of downstream task performance. To do so, we require nodes with flexible computational ability.

Advances in deep learning (DL) have produced many recurrent neural network (RNN) architectures that can be trained to perform a variety of downstream tasks. We thus use "graphs of RNNs", *i.e.* networks composed of message-passing RNNs as a generic substrate to show the following:

- **Noise can cause "cascading failure" in network computation.** By injecting noise, task performance easily degrades past the level achieved when only dropping out nodes.
- **Dropout [6] minimally affects dynamical robustness.** Despite Dropout's usefulness in creating networks robust to input variations, we find it does not improve robustness against strong noise perturbations.
- **Node centrality weakly correlates with performance degradation.** Noise in distant nodes degrades task performance similarly to noise in central nodes.

Taken together, these findings show dynamical robustness is a distinct characteristic from structural robustness that merits further investigation in computational networks.

2 Related Work

Though the specific questions described in this work are rather niche, there are many adjacent topics in dynamical robustness. The two primary relevant topics are dynamical robustness in network science, and regularizing robustness in deep learning.

Dynamical robustness in network science is defined as the ability of the network to maintain its activity when parts of the network are depressed or perturbed [1]. Dynamical robustness has two distinctions from structural robustness. First, it measures dynamical activity, e.g. activation levels or synchronization, rather than structural properties. In particular, in the context of computational networks, we are interested in the ability of the network to support computation, *i.e.* task performance. Second, the attack is the perturbation, rather than the removal of nodes. *e.g.*, Alstott et al. [7] studies brain lesioning effects on brain dynamics, but does so by deleting nodes rather than perturbing them. Joyce et al. [8] studied how simulated lesions of brain areas changes dynamical activity with respect to different network and node properties like centrality. In these examples, dynamical failure is modeled as a binary event. While node failure and removal clearly affects network topology and thus computation [9], less is known about how perturbations can be absorbed or distributed into existing dynamical activity. Tanaka et al. [1] shows low-degree nodes can play out-sized roles in network dynamics in contrast with structural dependence on high degree nodes. Later, He et al. [3] show the conditions for that finding are limited. In general, dynamical robustness appears to be dependent on not just network toplogy, but also the rich node dynamics [4].

Robustness in deep learning is typically evaluated in terms of performance against adversarial inputs, or expressing the ability of a model to generalize to out-of-domain data. Differently, we are interested in robustness in the course of computation. The view of deep networks as dynamical systems is a newly active research area, and so studies of modifications to internal state are limited. Maheswaranathan and Sussillo [10] explores the dynamics of an RNN in sentiment analysis, e.g. showing principled perturbation of hidden state can act as a sentiment negator. Some preliminary work (less chaotic, bistable) at the intersection of dynamical systems theory and deep learning propose RNN variants that are less chaotic, or reliant on a well-known dynamical system. In general, the mathematical machinery from dynamical systems theory is not yet directly applicable to dynamical robustness.

The closest topic in deep learning to dynamical robustness is regularization, which adds techniques such as dropout [6] to reduce network co-dependencies, or noising activations [11] as an information-theoretic pressure for learning sparser representations. As mentioned, these techniques are typically evaluated on end task performance. This work will do the same, that is, rather than measuring network activity as in complex systems studies [8], we evaluate the ability of such activity to perform the original computation.

Modular networks in deep learning have been introduced in a few different contexts, mainly as a mechanism for generalization. For example, Goyal et al. [12] introduce a technique called RIMs that uses several sparsely interacting RNNs to generalize to out-of-domain inputs. Neural Module Networks [13] uses different neural network components for a compositional approach to Visual Question-Answering.

To the author's best knowledge, there is no highly related work on robustness at this intersection of deep learning and network science. As in [8], I believe this problem may grow in importance as the computational neuroscience community gains the tools to model the dynamics between brain regions [14].

3 Methods

We instantiate a generic computational network by letting each node be a GRU [15], meaning that our overall graph network is a Gated Graph Neural Network [16]. The larger network structure is defined by a random Erdos-Renyi sample $G(n, p)$.

Different downstream tasks will require different amounts of communication between the nodes. We consider three demonstrative tasks, presented in increasing order of required communication.

1. Sinusoid: At each timestep, each node increments a local position on a noised sinusoidal function. Each node receives t initial steps from the sinusoid, and predict the next T steps. We evaluate mean squared error for all T timesteps, for all nodes. This task only requires local computation.
2. Sequential MNIST (SeqMNIST) [12]: As a variant on the MNIST digit recognition task, the network is fed a sequence of MNIST pixels (each node is provided the step's pixel value, total of $14x14 = 196$ steps), and is asked to classify the digit fed. We max-pool the final node activations to evaluate accuracy at the final timestep. This task is known to benefit from global communication [12], though it does not require it.
3. Density Classification (DC) [8][17]: nodes are given a binary flag at $t = 0$. After T timesteps, each node predicts the flag that the majority of nodes received as input. The difficulty of this task depends on the distribution of initial conditions (the IC). As in Mitchell and Crutchfield [17], we characterize the IC by a sampling each node's initial state from a Bernoulli distribution $\sim$ Ber(p) for each node's input. We train using a dataset generated with $p \sim U[0, 1]$, but evaluate on a more challenging "unbiased" condition, $p = 0.5$. Differently from Mitchell and Crutchfield [17], we evaluate accuracy of a network by averaging node accuracies, instead of requiring a single network prediction. Should we require all nodes to make the same prediction, we could not distinguish local and spreading failures on perturbation. We evaluate per-node accuracy at the final timestep. This task requires global communication, without much local computation.

For all models, we add minimal training bells and whistles. The architecture is constant across tasks excepted for a readout layer. For the density classification task, we add an edge embedding to each message, to allow a node to distinguish which neighbor is sending information. Networks achieve high performances under a variety of training hyperparameters, (though for consistency we fix them across tasks).

3.1 Noise injection

We are interested in modeling the spread of noise in the network. We model this by performing Gaussian Noise Injection (GNI) into the state of a select node n, at timestep $t \in [T]$. The noise is sampled $\sim N(0, \sigma^2)$, and is hereafter referred to as a "pulse" of strength σ, $P(\sigma)$. We compare the effects of GNI at different strengths, alongside the effects of silencing a node by zeroing its state, to represent the type of node failure studied in structural robustness. In order to have GNI have a constant effect with respect to the scale of model activations across tasks, we use Layer Normalization [18].

4 Results

We address the following questions:

1. **How does noise propagate?** We qualify whether noisy computation results in a different failure mode than inactivation (*i.e.* the traditional failure model).
2. **How can a model be made robust to noise?** Real-world networks are structurally robust to random dropout. We investigate whether Dropout [6] improves dynamical robustness.
3. **How does centrality of the perturbed node impact propagation?** Node centrality plays an important role in structural robustness. We ask what role it plays in dynamical robustness.

We consider G drawn from $G(N, p)$. (Though code is available to run the experiment across graphs and seeds, we only had the computational resources to evaluate a single draw.) Task information

Task	Nodes N	Timesteps T	Diam(G)	Metric	Performance (+ Dropout)
Sinusoid	12	20	3	MSE (↓)	0.111 (0.114)
Density Classification	149 [17]	21	5	Accuracy (↑)	0.948 (0.964)
Sequential MNIST	12 [12]	196	3	Accuracy (↑)	0.977 (0.979)

Table 1. Information about models in each task. Performance of models absent of noise is included, with and without dropout regularization. Arrows denote direction of improving performance.

is summarized in Fig. 1. We use task performance degradation (on the entire validation set) as the primary metric, as proxy for the degradation of the networks overall computational abilities. When performance is measured on a per-node basis, we exclude the perturbed nodes. Degradation is normalized by the original task performance, and equals 1.0 in DC and Sequential MNIST when the network performs at chance (there is no upper bound for degradation in the Sinusoid task). Degradation is measured as an average of 5 perturbation trials.

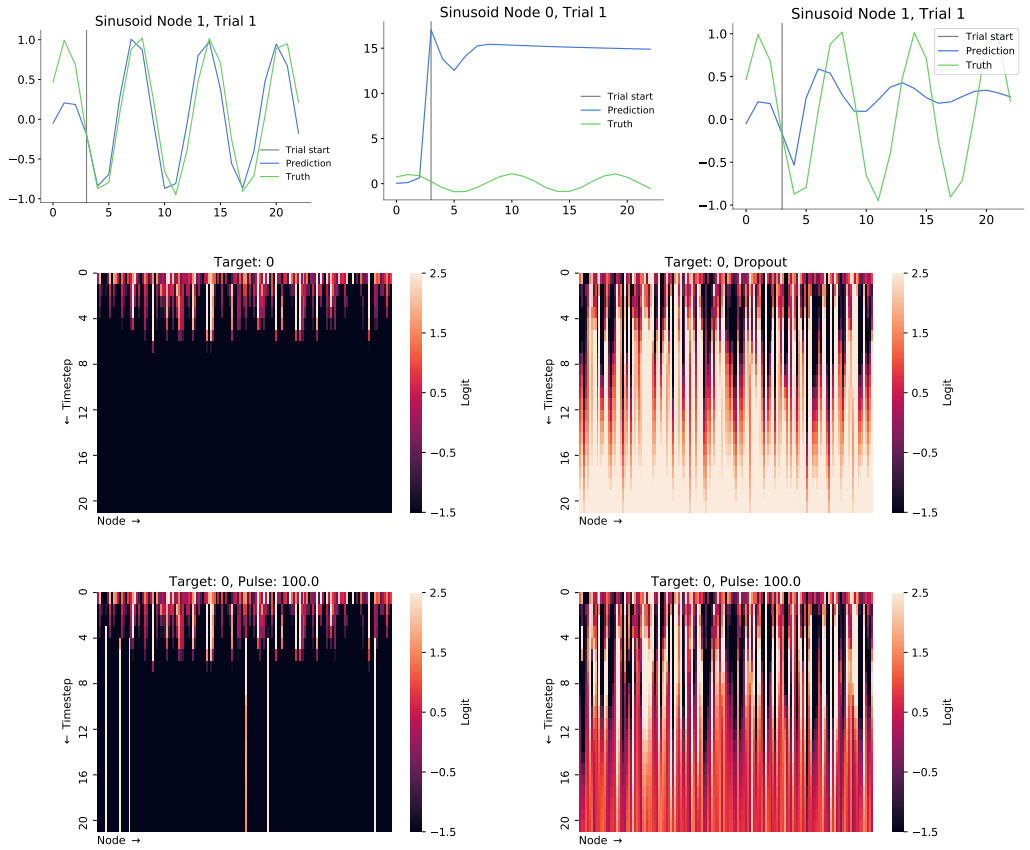

Figure 1. Noise can propagate in Sinusoid and DC. In the Sinusoid task (top), noising node 0 dampens the predictions of neighboring node 1. In the DC task (bottom), noising will occassionally only affect the noised nodes, and occassionally cascade beyond the affected nodes to change the whole network's predictions.

Noise propagates beyond the perturbed node. We show qualitative examples of GNI$\sim P(10)$ degrading performance beyond the targeted node in Fig. 1 for the Sinusoid and DC tasks. In the Sinusoid plot, we see (row 1, left) the model predictions closely match the true oscillation. However, when we pulse 1 node (Node 0), and see (row 1, center) that we have unrecoverably damaged the computational state. Moreover, the noise has propagated (row 1, right) to the neighboring node (Node 1), where the node predicted oscillations have been dampened, presumably meaning the nodes precise internal dynamics have been derailed. In the DC task, we first present an unnoised reference where the majority class is 0. Nodes quickly (in < 10 steps) converge to the majority decision (row 2, left).

When we add dropout (row 2, right), nodes take longer to converge, but are ultimately still able to compute properely. We then pulse 10 nodes, and see in some instances (row 3, left) only the perturbed nodes are affected, and the rest of the network converges unaffected, whereas in other instances (row 2, right) the network predictions diverge, where some nodes predict the original majority class 0 and others predict the incorrect class 1. These figures qualitatively demonstrate that noise affects neighboring nodes.

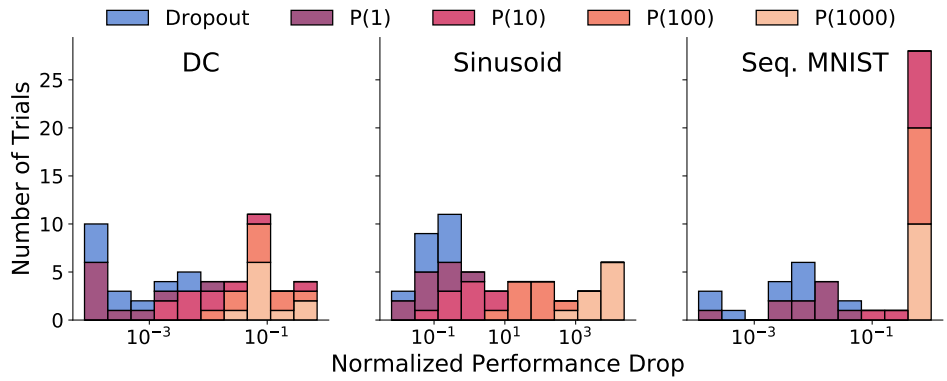

Figure 2. Histogram of pulse effects. Increasing pulse strengths increasingly degrade the model, outpacing Dropout, though results across tasks are varied.

Note that the effects of perturbation are affected by both the node(s) perturbed and the timestep of perturbation. To evaluate the effect of GNI quantitatively, we sample 10 random node-timestep pairs and apply perturbations of different strengths (on a logarithmic scale), as well as inactivation, which we denote as "Dropout". A distribution of the resulting effects are shown in Fig. 2. We note:

1. The effect of noise as the scale of module activations (*i.e.* 1.0) does not appear more harmful than dropout. That is, we do not find that noise can push nodes into "deceptive" states,that send misinformed messages to neighbors rather than implausible ones.

2. Both dropout and noise$\sim P(1)$ do not make a significant impact on model performance ($\sim 10^{-2}$ for the tasks where degradation is bounded, DC and Seq. MNIST).

3. The effect of logarithmically increasing noise has varying effects across tasks. In the Sinusoid task, where nodes are not expected to communicate, degradation also scales logarithmically. In Seq. MNIST, there's a clear cascade point, *i.e.* after GNI$\sim P(10)$, the network is unable to compute, presumably because precise internal states are key to the classification. The results in DC are in between, perhaps due to a reduced requirement for precise information communication.

The last point in particular suggests that the effect of noise (and vulnerability to dynamical cascades) scales with the amount of information passed between nodes. This could be investigated in a follow-up experiment using a family of tasks parameterized by the bandwidth of information required to succeed.

Dropout regularization does not reduce noise propagation. Note that simply inactivating an node (as in Perturbation) will prevent the node from solving the task. To achieve the same effect as Dropout for the sake of studying noise propagation, we implement Dropout regularization by randomly silencing 10% of message propagation in training. We run the same perturbation as in the previous section and obtain a new set of degradation results. We present the change in degradation (*i.e.* subtract original degradation from degradation with Dropout training) in Fig. 3. Here, we observe that training with Dropout largely does not effect the performance drop. The neutrality with respect to Dropout perturbation is likely due to the fact that degradation was already minimal (though it helps a bit *e.g.* in Seq. MNIST). Surprisingly, all tasks show a small trend where perturbation degradation increases when training with dropout. In some cases, like the Sinusoid task, the differences can be vast. It is unclear why this occurs in Sinusoid, but for the other tasks it is possible that in adding Dropout, the nodes actually place greater weight on the messages they do receive, and thus on any propagated noise.

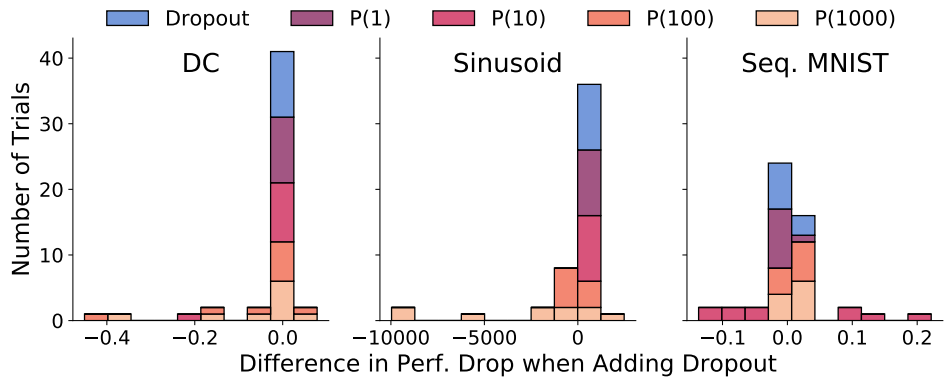

Figure 3. Dropout regularization has neutral effects on dynamical robustness.

Task	Degree	Eigenvalue	Katz	Harmonic	PageRank	Closeness	Betweenness
Sinusoid	−0.099	−0.082	−0.086	−0.068	−0.103	−0.046	**−0.143**
Density Classification	−0.153	**−0.168**	−0.166	−0.154	−0.147	−0.152	−0.14
Sequential MNIST	−0.134	−0.142	−0.138	−0.125	−0.131	−0.130	**−0.146**

Table 2. We evaluate Pearson correlation between degradation and node centrality. The most correlated centrality metric varies on the task, though the variance between metrics is not high.

4.1 Dynamical robustness is weakly correlated with node centralities

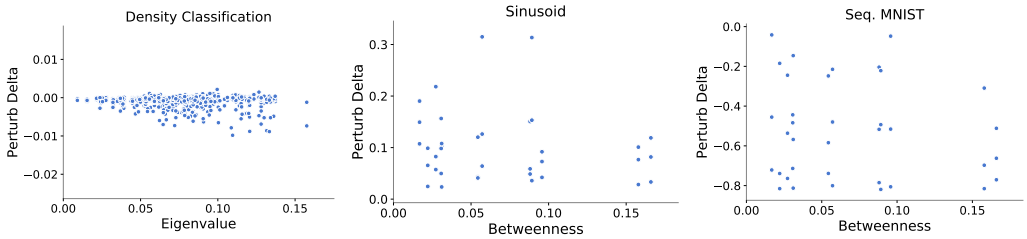

Figure 4. Relations between degradation and the most correlated centrality metric on each task. Even here, the correlation is weak.

We also find that the most strongly correlated centrality metric depends on the task. To do so, we perturb each node of a network on each task at an early, intermediate, and late timestep, and measured the Pearson correlation coefficient between the degradation resulting from a given node and its centrality metric. A full table of the resulting correlations are presented in Fig. 2 and the most strongly correlated metrics are presented in Fig. 4. Fig. 2 suggests even the most correlated metrics are weakly related with perturbation effects, and variance across metrics is also small. Nonetheless, metrics point in the direction we expect,*i.e*. perturbing more central nodes degrade the network further. There is some variation in the most strongly correlated metric, suggesting dynamical robustness is a heterogeneous problem, either dependent on the computational task at hand, or on the network structure (as both Sinusoid and Sequential MNIST share graph structure). The plot for degradation in DC Fig. 4 makes it clear a large majority of perturbations may have no effect. It is possible that in those cases, the majority of the noise energy is placed in unused message subspaces. These results do not paint a conclusive picture of the relationship between node centrality and dynamical robustness, and would likely be strengthened by considering results across a number of graph structures.

5 Discussion

Networks can achieve complex computation, given nodes with internal dynamics. We want to assess whether the standard abstraction of failure propagation is appropriate when perturbing the

activity of such complex networks. I show noise has potentially unbounded impact on computational processes occurring on networks, quickly outpacing the effects of attacking the network structure itself, *i.e*. by removing individual nodes. This motivates further study into the dynamical robustness of computational networks.

In future work, the author recommends identifying a computationally tractable way to characterize the effects of noising larger subsets of the network. Network cascades indeed rarely begin from the failure of only one node, and as I have shown noising as few as 3% of the nodes in the DC task can disable much of the network.

References

[1] G. Tanaka, K. Morino, and K. Aihara. Dynamical robustness in complex networks: the crucial role of low-degree nodes. *Scientific Reports*, 2(1):232, Jan 2012. ISSN 2045-2322. doi:10.1038/srep00232. URL https://doi.org/10.1038/srep00232. 1, 2

[2] C. E. Stafstrom and L. Carmant. Seizures and epilepsy: an overview for neuroscientists. *Cold Spring Harbor perspectives in medicine*, 5(6):a022426, Jun 2015. ISSN 2157-1422. doi:10.1101/cshperspect.a022426. URL https://pubmed.ncbi.nlm.nih.gov/26033084. 26033084[pmid]. 1

[3] Z. He, S. Liu, and M. Zhan. Dynamical robustness analysis of weighted complex networks. *Physica A: Statistical Mechanics and its Applications*, 392(18):4181 – 4191, 2013. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2013.05.005. URL http://www.sciencedirect.com/science/article/pii/S037843711300410X. 1, 2

[4] A. Buscarino, L. V. Gambuzza, M. Porfiri, L. Fortuna, and M. Frasca. Robustness to noise in synchronization of complex networks. *Scientific Reports*, 3(1):2026, Jun 2013. ISSN 2045-2322. doi: 10.1038/srep02026. URL https://doi.org/10.1038/srep02026. 1, 2

[5] L. V. Gambuzza, A. Buscarino, L. Fortuna, M. Porfiri, and M. Frasca. Analysis of dynamical robustness to noise in power grids. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 7(3): 413–421, 2017. 1

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html. 2, 3

[7] J. Alstott, M. Breakspear, P. Hagmann, L. Cammoun, and O. Sporns. Modeling the impact of lesions in the human brain. *PLoS computational biology*, 5(6):e1000408–e1000408, Jun 2009. ISSN 1553-7358. doi:10.1371/journal.pcbi.1000408. URL https://pubmed.ncbi.nlm.nih.gov/19521503. 19521503[pmid]. 2

[8] K. E. Joyce, S. Hayasaka, and P. J. Laurienti. The human functional brain network demonstrates structural and dynamical resilience to targeted attack. *PLoS computational biology*, 9(1):e1002885–e1002885, 2013. ISSN 1553-7358. doi:10.1371/journal.pcbi.1002885. URL https://pubmed.ncbi.nlm.nih.gov/23358557. 23358557[pmid]. 2, 3

[9] A.-L. Barabási and M. Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016. ISBN 9781107076266 1107076269. URL http://barabasi.com/networksciencebook/. 2

[10] N. Maheswaranathan and D. Sussillo. How recurrent networks implement contextual processing in sentiment analysis, 2020. 2

[11] B. Poole, J. Sohl-Dickstein, and S. Ganguli. Analyzing noise in autoencoders and deep networks. *CoRR*, abs/1406.1831, 2014. URL http://arxiv.org/abs/1406.1831. 2

[12] A. Goyal, A. Lamb, J. Hoffmann, S. Sodhani, S. Levine, Y. Bengio, and B. Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019. 2, 3, 4

[13] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Deep compositional question answering with neural module networks. *CoRR*, abs/1511.02799, 2015. URL http://arxiv.org/abs/1511.02799. 2

[14] M. G. Perich and K. Rajan. Rethinking brain-wide interactions through multi-region 'network of networks' models. *Current Opinion in Neurobiology*, 65:146 – 151, 2020. ISSN 0959-4388. doi: https://doi.org/10.1016/j.conb.2020.11.003. URL http://www.sciencedirect.com/science/article/pii/S0959438820301707. 2

[15] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014. 3

[16] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel. Gated graph sequence neural networks. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.05493. 3

[17] M. Mitchell and J. Crutchfield. G 1 . 15 : Computer science application : Evolving cellular automata to perform computations. 2001. 3, 4

[18] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. 3