Syntactic Learnability of Echo State Neural Language Models at Scale

Ryo Ueda¹ Tatsuki Kuribayashi² Shunsuke Kando¹ Kentaro Inui^{2,3,4}

¹The University of Tokyo ²MBZUAI ³Tohoku University ⁴RIKEN

{ryoryoueda, skando}@is.s.u-tokyo.ac.jp {Tatsuki.Kuribayashi, Kentaro.Inui}@mbzuai.ac.ae

Abstract

What is a neural model with minimum architectural complexity that exhibits reasonable language learning capability? To explore such a simple but sufficient neural language model, we revisit a basic reservoir computing (RC) model, Echo State Network (ESN), a restricted class of simple Recurrent Neural Networks. Our experiments showed that ESN with a large hidden state is comparable or superior to Transformer in grammaticality judgment tasks when trained with about 100M words, suggesting that architectures as complex as that of Transformer may not always be necessary for syntactic learning.

1 Introduction

If there exists a neural language model (LM) that has an architecture with minimum complexity yet has a language acquisition capability comparable to humans, what would it look like? What architecture would meet such a minimum requirement? To answer such scientific questions ultimately, we begin with a neural LM that is as simple as possible and evaluate its language acquisition capability, especially on syntactic generalization. Such a question can not directly be answered with the recent success of Transformer-based large LMs (LLMs) [Vaswani et al., 2017] alone, where the settings/architectures might be overly complex beyond cognitively-inspired motivations.

In this paper, we adopt a neural network class called *Echo State Network* [ESN, Jaeger, 2001, Lukoševičius and Jaeger, 2009, Rodan and Tino, 2011, Lukoševičius, 2012]. It is a special class of simple Recurrent Neural Networks in which only an output matrix is trainable, while the other parameters, e.g., recurrent matrix, are fixed (frozen) after initialization. Despite its highly restrictive nature, ESN has widely been used in time series processing and has demonstrated surprising capability [Tanaka et al., 2019]; for example, ESN served as a strong baseline for text classification tasks [Wieting and Kiela, 2019, Cabessa et al., 2021] as well as a subject of analyses in computational (psycho-)linguistic studies [Tong et al., 2007, Frank and Čerňanský, 2008, Frank and Bod, 2011].

In our experiments, we trained ESN-based LMs with about 100M words, to which human children are thought to be exposed by 13 years old, at a larger scale than the age ESN being actively explored [Tong et al., 2007]. We compared them with LSTM [Hochreiter and Schmidhuber, 1997] and Transformer LMs. Our experimental results demonstrate that ESN with a sizeable hidden state was comparable or superior to Transformer trained from scratch, while LSTM showed the best result in our setting, suggesting the benefits of gate mechanisms in the working memory.

Our results imply that one may not need overly complex neural network architectures and back-propagation through time to achieve a certain level of linguistic competence, encouraging more focus on simpler LMs in interdisciplinarily contextualizing the recent progress of neural LMs with the science of language. As more specific future work, it might also be worthwhile to investigate various topologies in the recurrent architecture for future research, as discussed in previous ESN studies [Deng and Zhang, 2007, Kawai et al., 2019].

2 Echo State Network

2.1 Definition and Initialization of ESN

Let $(\boldsymbol{u}_t)_{t=1}^T$ be an input sequence, where $\boldsymbol{u}_t \in \mathbb{R}^{N_{\text{vocab}}}$ (e.g., one-hot vector), and $\boldsymbol{h}_0 = \boldsymbol{0}$. ESN updates its state \boldsymbol{h}_t and output \boldsymbol{o}_t over time step $t = 0, \dots, T$ as follows:

$$h_{t+1} = (1 - a) \odot h_t + a \odot f(W_{\text{rec}}h_t + W_{\text{in}}u_{t+1}),$$

$$o_{t+1} = W_{\text{out}}h_{t+1} + b_{\text{out}},$$
(1)

where $W_{\text{in}} \in \mathbb{R}^{N_{\text{state}} \times N_{\text{vocab}}}$ is an input matrix, $W_{\text{rec}} \in \mathbb{R}^{N_{\text{state}} \times N_{\text{state}}}$ is a recurrent matrix, $W_{\text{out}} \in \mathbb{R}^{N_{\text{vocab}} \times N_{\text{state}}}$ is an output bias, $a \in \mathbb{R}^{N_{\text{state}}}$ is a leaking rate, f is an element-wise activation function such as tanh, and \odot is the element-wise product. W_{out} and b_{out} are trainable, whereas W_{in} , W_{rec} , and a are frozen.

Initialization of W_{in} First, we randomly generate matrices M_{in} and V_{in} of the same shape as the input matrix via element-wise i.i.d sampling: $(M_{in})_{ij} \stackrel{\text{i.i.d}}{\sim} \text{Bernoulli}(\gamma)$ and $(V_{in})_{ij} \stackrel{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma_{in}^2)$. Then, we obtain the input matrix as:

$$W_{\rm in} = M_{\rm in} \odot V_{\rm in}, \tag{2}$$

where $\sigma_{\rm in} > 0$ is an *input scale* and $\gamma \in (0,1]$ is a *connectivity*. The benefit of introducing sparse connectivity to $\mathbf{W}_{\rm in}$ is discussed in Gallicchio [2020].

Initialization of W_{rec} Similarly to the input matrix initialization, we randomly generate matrices M_{rec} and V_{rec} as: $(M_{\text{rec}})_{ij} \stackrel{\text{i.i.d}}{\sim} \text{Bernoulli}(\gamma)$ and $(V_{\text{rec}})_{ij} \stackrel{\text{i.i.d}}{\sim} \mathcal{N}(0,1)$. Then, we obtain the recurrent matrix as follows:

$$W_{\text{rec}} = \frac{\rho_{\text{rec}}}{\rho(M_{\text{rec}} \odot V_{\text{rec}})} M_{\text{rec}} \odot V_{\text{rec}}, \tag{3}$$

where $\rho(\cdot)$ indicates the *spectral radius* of a given matrix, and $\rho_{\rm rec} > 0$ specifies the spectral radius of $\boldsymbol{W}_{\rm rec}$. In general, the spectral radius of a matrix \boldsymbol{M} is defined as the largest value out of the absolute eigenvalues of \boldsymbol{M} , which intuitively indicates to what extent \boldsymbol{M} can expand a vector \boldsymbol{v} by $\boldsymbol{M}\boldsymbol{v}$. The ESN model's performance is empirically (and to some extent theoretically) known to be maximized in most cases if $\rho_{\rm rec}$ is sufficiently close to but never exceeds 1 [Jaeger, 2001]. If $\rho_{\rm rec}$ is too small, the state \boldsymbol{h}_t forgets the past input sequence too quickly. if $\rho_{\rm rec}$ is too large, \boldsymbol{h}_t evolves so chaotically that the generalization becomes almost impossible.

Initialization of a The leaking rate vector a is initialized as:

$$(\boldsymbol{a})_i \stackrel{\text{i.i.d}}{\sim} \text{Uniform}(\alpha_{\min}, \alpha_{\max}),$$
 (4)

where $\alpha_{\rm min}$ and $\alpha_{\rm max}$ are hyperparameters specifying the minimum and maximum leaking rates respectively. Although, in a typical formulation, the leaking rate is a scalar hyperparameter (or equivalently, just setting $\alpha_{\rm min} = \alpha_{\rm max}$), the vectorized formulation is known to yield more complex, multi-scale dynamics [Tanaka et al., 2022]. We expect it to give ESN a desirable language modeling property, recalling the gate mechanism's empirical success in LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Cho et al., 2014].

Parameterization of W_{out} The output matrix is usually parametrized as a full-rank dense matrix, but we parametrize it via low-rank decomposition to reduce the parameters, i.e., $W_{\text{out}} = AB$, where $A \in \mathbb{R}^{N_{\text{vocab}} \times r_{\text{out}}}$ and $B \in \mathbb{R}^{r_{\text{out}} \times N_{\text{state}}}$ are trainable matrices, with some $r_{\text{out}} < \min\{N_{\text{state}}, N_{\text{vocab}}\}$. We summarize ESN's hyperparameters in table 1 and parameters in table 2.

 $^{^{1}\}boldsymbol{A}, \ \boldsymbol{B}, \ \mathrm{and} \ \boldsymbol{b}_{\mathrm{out}} \ \mathrm{are \ initialized \ with \ the \ PyTorch \ module \ (nn.Linear):} \ (\boldsymbol{A})_{ij}, \ (\boldsymbol{b}_{\mathrm{out}})_{i} \overset{\mathrm{i.i.d}}{\sim} \mathrm{Uniform}(-\sqrt{r_{\mathrm{out}}^{-1}}, \sqrt{r_{\mathrm{out}}^{-1}})$ and $(\boldsymbol{B})_{ij} \overset{\mathrm{i.i.d}}{\sim} \mathrm{Uniform}(-\sqrt{N_{\mathrm{state}}^{-1}}, \sqrt{N_{\mathrm{state}}^{-1}}).$

Hyperparameter name	Math symbol
state size	$N_{ m state}$
spectral radius	$ ho_{ m rec}$
input scale	$\sigma_{ m in}$
connectivity	$\gamma \ (= d/N_{\rm state})$
minimum leaking rate	$lpha_{ m min}$
maximum leaking rate	$\alpha_{ m max}$
activation function	f
output rank	$r_{ m out}$

Table 1: Hyperparameters of ESN, adopted in this paper.

Parameter name	Math symbol	Frozen	Relevant hyper- parameter(s)
input matrix	$oldsymbol{W}_{ m in}$	√	$N_{\mathrm{state}}, \gamma, \sigma_{\mathrm{in}}$
recurrent matrix	$W_{ m rec}$	√	$N_{\mathrm{state}}, \gamma, \rho_{\mathrm{rec}}$
leaking rate(s)	a	✓	$N_{\rm state}, \alpha_{\rm min}, \alpha_{\rm max}$
output matrix	$W_{ m out}$		$N_{ m state}, r_{ m out}$
output bias	$oldsymbol{b}_{ ext{out}}$		$r_{ m out}$

Table 2: Parameters of ESN, adopted in this paper.

2.2 ESN as Language Model

Let $s = (w_t)_{t=1}^T$ be an input sentence, where each $w_t \in \{1, \dots, N_{\text{vocab}}\}$ indicates a token (word) index. The first token w_t is the special symbol indicating the beginning-of-sentence (BOS), while the last token w_t is the special token indicating the end-of-sentence (EOS). Following the standard practice in neural LMs, we obtain an input sequence $(u_t)_{t=1}^T$ by converting each index w_t into the corresponding one-hot vector $u_t \in \mathbb{R}^{N_{\text{vocab}}}$. The model log-probability of s is given by:

$$\log p(s \mid \boldsymbol{\theta}_{\mathrm{Tr}}, \boldsymbol{\theta}_{\mathrm{Fr}}) = \sum_{t=1}^{T-1} \log p(w_{t+1} \mid (w_i)_{i=1}^t; \boldsymbol{\theta}_{\mathrm{Tr}}, \boldsymbol{\theta}_{\mathrm{Fr}}) = \sum_{t=1}^{T-1} \log \left(\operatorname{softmax}(\boldsymbol{o}_t) \right)_{w_{t+1}}, \quad (5)$$

where θ_{Tr} and θ_{Fr} are vectorizations of trainable and frozen parameters, respectively.

2.3 Number of Parameters in ESN

In this section, we discuss the number of parameters in ESN. We only count nonzero components as parameters and assume highly sparse connectivity, i.e., $\gamma = d/N_{\rm state}$ for some $d \ll N_{\rm state}$. First, since $\mathbf{W}_{\rm in}$ is a $N_{\rm state} \times N_{\rm vocab}$ -matrix with connectivity γ , we have

$$\#(\mathbf{W}_{\rm in}) = N_{\rm state} N_{\rm vocab} \gamma = N_{\rm vocab} d, \tag{6}$$

where $\#(\cdot)$ denotes the (expected) number of parameters. Likewise, since \mathbf{W}_{rec} is a $N_{state} \times N_{state}$ -matrix with connectivity γ , we have

$$\#(\mathbf{W}_{\text{rec}}) = N_{\text{state}}^2 \gamma = N_{\text{state}} d. \tag{7}$$

As W_{out} is parametrized via low-rank decomposition, we have

$$\#(\mathbf{W}_{\text{out}}) = (N_{\text{vocab}} + N_{\text{state}})r_{\text{out}}.$$
 (8)

Also, $\#(\boldsymbol{a}) = N_{\text{state}}$ and $\#(\boldsymbol{b}_{\text{out}}) = N_{\text{vocab}}$. To sum up, the numbers of frozen $\boldsymbol{\theta}_{\text{Fr}}$, trainable $\boldsymbol{\theta}_{\text{Tr}}$, and total parameters $\boldsymbol{\theta}_{\text{All}}$ are

$$\#(\boldsymbol{\theta}_{\mathrm{Fr}}) = (N_{\mathrm{state}} + N_{\mathrm{vocab}})d + N_{\mathrm{state}},$$

$$\#(\boldsymbol{\theta}_{\mathrm{Tr}}) = (N_{\mathrm{state}} + N_{\mathrm{vocab}})r_{\mathrm{out}} + N_{\mathrm{vocab}},$$

$$\#(\boldsymbol{\theta}_{\mathrm{All}}) = (N_{\mathrm{state}} + N_{\mathrm{vocab}})(d + r_{\mathrm{out}} + 1).$$
(9)

Note that they grow only linearly with respect to N_{state} , which enables efficient scaling.

LM name	$N_{ m state}$	$\#(\boldsymbol{\theta}_{\mathrm{Tr}})$ [M]	$\#(oldsymbol{ heta}_{ m All}) \ [{ m M}]$	Train NLL↓ (per token)	Validation NLL↓ (per token)	BLiMP↑ [%]
ESN	1,024	26	28	$5.311 (\pm 0.003)$	$5.024 \ (\pm 0.003)$	$\overline{56.2 \ (\pm 0.3)}$
	2,048	27	29	$5.173 \ (\pm 0.002)$	$4.887 (\pm 0.001)$	$56.1 \ (\pm 0.3)$
	4,096	28	30	$5.071 \ (\pm 0.005)$	$4.794 (\pm 0.011)$	$57.9 \ (\pm 0.4)$
	8,192	30	32	$4.992 \ (\pm 0.005)$	$4.708 (\pm 0.011)$	$58.5 \ (\pm 0.3)$
	16,384	34	36	$4.965\ (\pm0.006)$	$4.642 \ (\pm 0.008)$	$59.2 \ (\pm 0.4)$
	32,768	43	45	$4.987 \ (\pm 0.023)$	$4.630 \ (\pm 0.008)$	$60.0 \ (\pm 0.3)$
	$65,\!536$	59	63	$5.102 \ (\pm 0.023)$	$4.690\ (\pm0.023)$	$60.5~(\pm 0.2)$
GPT2 Scratch	768	124	124	$5.667 (\pm 0.024)$	$4.803 (\pm 0.011)$	$\overline{58.7 (\pm 0.7)}$
LSTM	512	54	54	$4.503 \ (\pm 0.000)$	$4.120\ (\pm0.001)$	$67.8 \ (\pm 0.2)$
GPT2 OpenAI	768	124	124	-	-	82.2

Table 3: The number of trainable parameters $\#(\theta_{Tr})$, the total number of parameters $\#(\theta_{All})$, the train NLL (per token), the validation NLL (per token), and the overall BLiMP score [%]. The train NLLs are higher than the validation NLLs because the former are averaged over batches during training, while the latter are computed after one epoch. ($\pm \cdot$) represents one standard error of mean, computed from 4 runs for each configuration.

3 Why ESN?

In this section, we discuss again, but in more detail, why we should revisit ESN despite the era of LLMs. As a matter of computational linguistics (CL), rather than engineering NLP directions, one would seek minimum conditions (aka. Occam's razor) for some linguistic phenomena emerging. In this sense, there is no necessity to begin with a complex neural network architecture like Transformer. We should note that modern deep learning (e.g., Transformer) is a collection of heuristics whose counterparts in human brains are controversial; for example, humans do not seem (at least for now) to compute gradients by backpropagating the error, while deep learning history is considerably attributed to gradient stabilization, e.g., gradient clipping [Pascanu et al., 2013], long-short term memory [Hochreiter and Schmidhuber, 1997], Xavier/Kaiming initialization [Glorot and Bengio, 2010, He et al., 2015], residual connection [He et al., 2016], LayerNorm [Ba et al., 2016], and learning rate scheduling [Goyal et al., 2018]. Consequently, such complex designs may hinder the clear interpretation of its implication and what the model actually does in the study of CL, or at least we have to be aware of the potential risk of persisting with the particular architecture discovered by the narrow advancement of engineering-sided trial-and-errors, which may perhaps be too complex to progress the science of language.²

ESN can, in contrast, be regarded as a natural discretization of a simple, continuous-time neuronal model [Jaeger et al., 2007], which is suitable for exploring the minimum complexity. Indeed, ESN is a linear regression model and thus free from backpropagation through time (BPTT) and layers. More specifically, by regarding the update equation of the state as a feature function $\phi_{\theta_{\text{Fr}}}(\cdot)$ from a (prefix of) input sequence $(u_t)_{t=1}^{T'}$ to a state $h_{T'}$, one can restate ESN as a (log-)linear model $o_{T'} = W_{\text{out}}\phi_{\theta_{\text{Fr}}}\left((u_t)_{t=1}^{T'}\right) + b_{\text{out}}$. Previous studies adopted ESN in the context of computational (psycholinguistics a decade ago [Tong et al., 2007, Frank and Čerňanský, 2008, Frank and Bod, 2011], although they have now been replaced with modern deep learning methods [Gulordava et al., 2018, Wilcox et al., 2020]. Such a transition might be confounded by the advancement in optimization, scaling, or preprocessing techniques toward neural LMs rather than the essential differences in model architectures, and thus, it is worth revisiting ESN under modern settings. Notably, while the previous psycholinguistic works using ESN attempted at most $N_{\text{state}} \approx 1000$, we scale it up to 65, 536.³

4 Experiments

4.1 Experimental Setup

Dataset and Preprocessing We used the BabyLM Challenge dataset (2023 version) [Warstadt et al., 2023, Choshen et al., 2024] as the training and validation datasets. The training data consists of approx-

²Internal interpretation of Transformers, such as BERTology [e.g., Tenney et al., 2019] and mechanistic interpretability [e.g., Cunningham et al., 2023], is of course a promising and intriguing direction, considering the recent success of LLMs.

³PyTorch provides torch.sparse for efficient computation of sparse tensors. This allows us to scale up ESN to some extent, even with a laboratory-level GPU resource.

imately 100M words. Since they were provided as raw texts, we applied the NLTK sentence tokenizer⁴ and then applied the GPT2 tokenizer⁵. Thus, $N_{\text{vocab}} = 50257$. The maximum sequence length per sentence was set to 512, and sequences shorter than 6 were removed so that LMs would learn longer dependency structures.⁶

ESN's Configuration We set $\rho_{\rm rec} = 0.99$, $\sigma_{\rm in} = 1$, d = 32, $\alpha_{\rm min} = 0$, $\alpha_{\rm max} = 1$, $f(\cdot) = \tanh(\cdot)$, and $r_{\rm out} = 512$. We varied the state size as $N_{\rm state} \in \{2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}\}$, according to which $\gamma = d/N_{\rm state}$ also changed.⁷

Comparison Targets We compare ESNs with the GPT2 pre-trained by OpenAI (GPT2 OpenAI), a GPT2 trained from scratch (GPT2 Scratch), and an LSTM with an embedding and hidden state of size 512. GPT2 Scratch followed the default configuration of GPT2Config.⁸ We applied the dropout (p = 0.1) immediately after the embedding and before the output matrix application in LSTM.

Training Procedure The LMs were trained for just 1 epoch. The batch size was 32. AdamW [Loshchilov and Hutter, 2019] was used as an optimizer with its default parameters.⁹

Syntactic Evaluation To measure the extent to which the LMs accurately capture syntactic structure, we used BLiMP [Warstadt et al., 2020], a dataset of minimal pairs of English sentences covering various syntactic phenomena.

4.2 Results

Table 3 shows the number of training parameters $\#(\theta_{\rm Tr})$, the total number of parameters $\#(\theta_{\rm All})$, the train negative log-likelihood (NLL), the validation NLL, and the overall BLiMP score. Somewhat surprisingly, ESN outperforms GPT2 Scratch in the validation NLL when $N_{\rm state} \geq 4096$ and in the BLiMP score when $N_{\rm state} \geq 16384$, even though ESN has less trainable/total parameters than GPT2. The scaling generally improved the performance of ESN, like the scaling law [Kaplan et al., 2020], monotonically for BLiMP scores, while the train and validation NLL improved until $N_{\rm state} = 16384$; we suspect this is just due to the limited training time (i.e., 1 epoch). These tentatively suggest that large ESNs can also learn core syntactic phenomena. GPT2 OpenAI achieves the highest BLiMP score, which is not surprising for its pre-training scale, while this is still useful to grasp the "upper bound" of LMs at a similar scale. Excluding GPT2 OpenAI, LSTM exhibits the best results in the train NLL, validation NLL, and BLiMP; the inferiority of the Transformer to LSTM might reflect the transformer's data inefficiency in the BabyLM setting.

5 Conclusion

This paper briefly revisited a basic reservoir computing model called Echo State Network (ESN) as a neural language model. The experimental results showed that ESN is comparable to or superior to the Transformer model, at least on the BabyLM scale. The best result of LSTM suggests that it is worth revisiting the benefit of gate mechanisms as well. Exploring better topologies in the ESN's architecture will also be worthwhile beyond the simple, sparse connectivity adopted in this paper.

Acknowledgments

This work was supported by JSPS KAKENHI (Grant Number JP23KJ0768) and JST ACT-X (Grant Number JPMJAX24C5).

⁴https://www.nltk.org/api/nltk.tokenize.sent_tokenize.html

 $^{^5 \}texttt{https://huggingface.co/docs/transformers/model_doc/gpt2\#transformers.GPT2Tokenizer}$

⁶Minimum sentence length should be 4 due to BOS/EOS.

⁷We also investigated the influence of the connectivity and leaking rates. See Appendices A and B respectively.

⁸https://huggingface.co/docs/transformers/model_doc/gpt2#transformers.GPT2Config

⁹https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html

References

- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.
- Jérémie Cabessa, Hugo Hernault, Heechang Kim, Yves Lamonato, and Yariv Z. Levy. Efficient text classification with echo state networks. In 2021 International Joint Conference on Neural Networks (IJCNN), pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533958.
- Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST@EMNLP 2014*, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014, pages 103–111. Association for Computational Linguistics, 2014. doi: 10.3115/V1/W14-4012. URL https://aclanthology.org/W14-4012/.
- Leshem Choshen, Ryan Cotterell, Michael Y. Hu, Tal Linzen, Aaron Mueller, Candace Ross, Alex Warstadt, Ethan Wilcox, Adina Williams, and Chengxu Zhuang. [call for papers] the 2nd babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus. CoRR, abs/2404.06214, 2024. doi: 10.48550/ARXIV.2404.06214. URL https://doi.org/10.48550/arXiv.2404.06214.
- Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.
- Zhidong Deng and Yi Zhang. Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Trans. Neural Networks*, 18(5):1364–1375, 2007. doi: 10.1109/TNN.2007.894082. URL https://doi.org/10.1109/TNN.2007.894082.
- Stefan L. Frank and Rens Bod. Insensitivity of the human sentence-processing system to hierarchical structure. *Psychological Science*, 22(6):829–834, 2011. doi: 10.1177/0956797611409589. URL https://doi.org/10.1177/0956797611409589. PMID: 21586764.
- Stefan L. Frank and Michal Čerňanský. Generalization and systematicity in echo state networks. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 30(30), 2008.
- Claudio Gallicchio. Sparsity in reservoir computing neural networks, 2020. URL https://arxiv.org/abs/2006.02957.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.
- Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018. URL https://arxiv.org/abs/1706.02677.
- Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics, 2018. doi: 10.18653/V1/N18-1108. URL https://doi.org/10.18653/v1/n18-1108.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.
- Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology GMD Technical Report 148, 2001. Erratum note available at https://www.ai.rug.nl/minds/uploads/EchoStatesTechRepErratum.pdf.
- Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2007.04.016. URL https://www.sciencedirect.com/science/article/pii/S089360800700041X. Echo State Networks and Liquid State Machines.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.
- Yuji Kawai, Jihoon Park, and Minoru Asada. A small-world topology enhances the echo state property and signal propagation in reservoir computing. *Neural Networks*, 112:15–23, 2019. doi: 10.1016/J. NEUNET.2019.01.002. URL https://doi.org/10.1016/j.neunet.2019.01.002.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.
- Mantas Lukoševičius. *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_36. URL https://doi.org/10.1007/978-3-642-35289-8_36.
- Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3):127-149, 2009. ISSN 1574-0137. doi: https://doi.org/10.1016/j.cosrev.2009.03.005. URL https://www.sciencedirect.com/science/article/pii/S1574013709000173.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2013. URL https://arxiv.org/abs/1211.5063.
- Ali Rodan and Peter Tino. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22(1):131–144, 2011. doi: 10.1109/TNN.2010.2089641.
- Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. Neural Networks, 115:100–123, 2019. doi: 10.1016/J.NEUNET.2019.03.005. URL https://doi.org/10.1016/j.neunet.2019.03.005.
- Gouhei Tanaka, Tadayoshi Matsumori, Hiroaki Yoshida, and Kazuyuki Aihara. Reservoir computing with diverse timescales for prediction of multiscale dynamics. *Phys. Rev. Res.*, 4:L032014, Jul 2022. doi: 10. 1103/PhysRevResearch.4.L032014. URL https://link.aps.org/doi/10.1103/PhysRevResearch.4.L032014.
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings* of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, pages 4593–4601. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1452. URL https://doi.org/10.18653/v1/p19-1452.
- Matthew H. Tong, Adam D. Bickett, Eric M. Christiansen, and Garrison W. Cottrell. Learning grammatical structure with echo state networks. *Neural Networks*, 20(3):424–432, 2007. doi: 10.1016/J.NEUNET.2007.04.013. URL https://doi.org/10.1016/j.neunet.2007.04.013.

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Trans. Assoc. Comput. Linguistics*, 8:377–392, 2020. doi: 10.1162/TACL_A_00321. URL https://doi.org/10.1162/tacl_a_00321.
- Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. Call for papers the babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus. CoRR, abs/2301.11796, 2023. doi: 10.48550/ARXIV.2301.11796. URL https://doi.org/10.48550/arXiv.2301.11796.
- John Wieting and Douwe Kiela. No training required: Exploring random encoders for sentence classification. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. URL https://openreview.net/forum?id=BkgPajAcY7.
- Ethan Wilcox, Jon Gauthier, Jennifer Hu, Peng Qian, and Roger Levy. On the predictive power of neural language models for human real-time comprehension behavior. In *Proceedings of the 42th Annual Meeting of the Cognitive Science Society Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020, virtual, July 29 August 1, 2020.* cognitivesciencesociety.org, 2020. URL https://cogsci.mindmodeling.org/2020/papers/0375/index.html.

A Influence of Sparse Connectivity

LM name	$N_{ m state}$	γ	$\#(\boldsymbol{\theta}_{\mathrm{Tr}})$ [M]	$\#(oldsymbol{ heta}_{ m All}) \ [{ m M}]$	Train NLL↓ (per token)	Validation NLL↓ (per token)	BLiMP↑ [%]
ESN	4,096	2^{-12}	28	28	$5.792 (\pm 0.027)$	$5.497 (\pm 0.027)$	$\overline{55.3 (\pm 0.7)}$
		2^{-11}		28	$5.387 (\pm 0.036)$	$5.080\ (\pm0.037)$	$56.1 \ (\pm 0.4)$
		2^{-10}		28	$5.137 (\pm 0.018)$	$4.833 \ (\pm 0.017)$	$56.8 \ (\pm 0.6)$
		2^{-9}		28	$5.063\ (\pm0.005)$	$4.761 \ (\pm 0.002)$	$58.3 \ (\pm 0.3)$
		2^{-8}		29	$5.055 (\pm 0.006)$	$4.756 \ (\pm 0.002)$	$56.9 \ (\pm 0.3)$
		2^{-7}		30	$5.071\ (\pm0.005)$	$4.794 (\pm 0.011)$	$57.9 (\pm 0.4)$
		2^{-6}		31	$5.094\ (\pm0.001)$	$4.820 \ (\pm 0.017)$	$57.6 \ (\pm 0.5)$
		2^{-5}		35	$5.141\ (\pm0.003)$	$4.836\ (\pm0.005)$	$56.7 (\pm 0.1)$
		2^{-4}		42	$5.178 (\pm 0.004)$	$4.890\ (\pm0.014)$	$57.7 \ (\pm 0.3)$
		2^{-3}		56	$5.228 \ (\pm 0.001)$	$4.943\ (\pm0.011)$	$56.9 \ (\pm 0.5)$
		2^{-2}		84	$5.286 \ (\pm 0.001)$	$4.994\ (\pm0.015)$	$56.6 \ (\pm 0.3)$
		2^{-1}		139	$5.350 \ (\pm 0.001)$	$5.041\ (\pm0.008)$	$56.7 \ (\pm 0.2)$
		1		251	$5.417 \ (\pm 0.001)$	$5.102\ (\pm0.008)$	$56.3 \ (\pm 0.3)$

Table 4: The number of trainable parameters $\#(\theta_{Tr})$, the total number of parameters $\#(\theta_{All})$, the train NLL (per token), the validation NLL (per token), and the overall BLiMP score [%], when the connectivity γ is varied. The train NLLs are higher than the validation NLLs because the former are averaged over batches during training, while the latter are computed after one epoch. (\pm ·) represents one standard error of mean, computed from 4 runs for each configuration.

We conducted an additional experiment to investigate the influence of sparse connectivity, as the high sparsity $\gamma \ll 1$ adopted in this paper is not a common practice in deep learning-based language modeling. We set $\rho_{\rm rec}=0.99$, $\sigma_{\rm in}=1$, $\alpha_{\rm min}=0$, $\alpha_{\rm max}=1$, $f(\cdot)=\tanh(\cdot)$, and $r_{\rm out}=512$, following the configuration of the main experiment. In this section, however, we fixed $N_{\rm state}=2^{12}=4096$ while varied d as $d\in\{1,2,2^2,2^3,2^4,2^5(=32),2^6,2^7,2^8,2^9,2^{10},2^{11},2^{12}\}$, i.e., $\gamma\in\{2^{-12},2^{-11},2^{-10},2^{-9},2^{-8},2^{-7},2^{-6},2^{-5},2^{-4},2^{-3},2^{-2},2^{-1},1\}$. The results are shown in Table 4. More sparse connectivity results in better validation NLL score until $\gamma=2^{-8}$, whereas further sparsity degrades the score. The BliMP scores show a similar tendency. They suggest surprisingly that the (appropriate) sparse connectivity benefits not only the efficient scaling but also the capability of language.

B Influence of Leaking Rates

LM name	$N_{ m state}$	α_{\min}	$\#(\boldsymbol{\theta}_{\mathrm{Tr}})$ [M]	$\#(oldsymbol{ heta}_{ m All}) \ [{ m M}]$	Train NLL↓ (per token)	Validation NLL↓ (per token)	BLiMP↑ [%]
ESN	4,096	0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9	28	30	$\begin{array}{c} 5.071 \ (\pm 0.005) \\ 5.093 \ (\pm 0.007) \\ 5.105 \ (\pm 0.006) \\ 5.120 \ (\pm 0.005) \\ 5.136 \ (\pm 0.004) \\ 5.155 \ (\pm 0.004) \\ 5.177 \ (\pm 0.004) \\ 5.202 \ (\pm 0.004) \\ 5.233 \ (\pm 0.004) \\ 5.272 \ (\pm 0.005) \\ 5.330 \ (\pm 0.006) \end{array}$	$4.794 (\pm 0.011)$ $4.803 (\pm 0.014)$ $4.823 (\pm 0.015)$ $4.829 (\pm 0.009)$ $4.845 (\pm 0.007)$ $4.861 (\pm 0.006)$ $4.884 (\pm 0.006)$ $4.909 (\pm 0.006)$ $4.936 (\pm 0.005)$ $4.971 (\pm 0.004)$ $5.018 (\pm 0.005)$	$\begin{array}{c} 57.9 \; (\pm 0.4) \\ 57.7 \; (\pm 0.3) \\ 57.6 \; (\pm 0.5) \\ 57.8 \; (\pm 0.4) \\ 57.7 \; (\pm 0.4) \\ 57.5 \; (\pm 0.4) \\ 57.4 \; (\pm 0.4) \\ 57.4 \; (\pm 0.4) \\ 57.2 \; (\pm 0.4) \\ 57.1 \; (\pm 0.3) \\ 56.8 \; (\pm 0.2) \\ \end{array}$

Table 5: The number of trainable parameters $\#(\theta_{Tr})$, the total number of parameters $\#(\theta_{All})$, the train NLL (per token), the validation NLL (per token), and the overall BLiMP score [%], when the minimum leaking rate α_{min} is varied. The train NLLs are higher than the validation NLLs because the former are averaged over batches during training, while the latter are computed after one epoch. (\pm ·) represents one standard error of mean, computed from 4 runs for each configuration.

We conducted another experiment to investigate the influence of leaking rates. We set $\rho_{\rm rec}=0.99$, $\sigma_{\rm in}=1,\ d=32,\ \alpha_{\rm max}=1,\ f(\cdot)=\tanh(\cdot),\ {\rm and}\ r_{\rm out}=512,$ following the configuration of the main experiment. In this section, however, we fixed $N_{\rm state}=2^{12}=4096$ while varied $\alpha_{\rm min}$ as $\alpha_{\rm min}\in\{0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1\}$. The results are shown in Table 5. Lower $\alpha_{\rm min}$ results in lower NLLs and higher BLiMP scores, which suggests that the multiple time scales indeed benefit the capability of language.