Forecasting of Spatio-temporal Chaotic Dynamics with Self-Attention

Colin Shea-Blymyer
Oregon State University
sheablyc@oregonstate.edu

Sizhen Li
Oregon State University
lisiz@oregonstate.edu

Abstract

Forecasting chaotic sequences is a difficult task for traditional machine learning techniques as small errors in approximation quickly lead to dramatic divergence. Recent work has shown that recurrent neural networks (RNNs) and reservoir computers (RCs) can effectively predict spatio-temporal systems. We extend this work by implementing a attention-based approach to dynamics forecasting, and quantify its relative prediction accuracy on the forecasting of the Kuramoto-Sivashinsky (KS) equation. We compare this method's performance and limitations to others that rely on recurrent connections. We find that the attention-based model's capability to incorporate long-range dependencies makes it a good fit for predicting complex dynamics, but it lacks the speed and efficiency of RC models. This study confirms the capability of self-attention architectures to capture chaotic dynamics at least as well as architectures with recurrent connections.

1. Introduction

Many important real-world systems exhibit spatio-temporal chaotic dynamics, e.g. fluid turbulence and the movement of electrical signals in the heart. Indeed, the seminal event that led Edward Lorenz to found the field of chaos theory was his discovery that a change from 6-digit precision to 3-digit precision would drastically change his weather pattern simulations [5]. From this, he determined that weather modeling could not make long-term forecasts [12]. This helps illustrate how difficult chaotic dynamics are to learn — small amounts of error in a model can lead to outcomes that diverge from the ground truth significantly. This also helps to show how important a problem the effective modeling of such systems is — from weather to cardiology, an accurate understanding of such systems is a crucial component to our understanding of the world.

Deep learning (recurrent neural networks, in particular) has greatly advanced the state of the art of system modeling [11]. These techniques can learn to model very complex systems with great accuracy and great generality. In fact, recent work has shown that even chaotic systems can now be forecasted beyond traditional limits of predictability [13]. However, these learned models still have significant drawbacks regarding their long-range effectiveness, training and memory complexity, and hyper-parameter selection process.

In this work, we extend a recent effort to compare the effectiveness of modeling chaos with different deep learning architectures [15]. Specifically, we investigate the effectiveness of an attention-based architecture [14] at this same task, and contrast it with recurrent architectures. Attention architectures outperform recurrent architectures in other sequence prediction tasks, so it is prudent to benchmark the approach against other popular techniques on this uniquely difficult task.

The next section reviews the mechanics of various recurrent and attention architectures and the use of deep learning for the prediction of chaotic systems. Section 3 describes how we implement two architectures (one recurrent, one attention) for testing, and section 4 concerns how these architectures were tested, evaluated, and provides results. Section 5 concludes the paper with a summary of our results, and a discussion of future work.

2. Background

Recurrent neural networks [4] (RNN) are a popular class of models that have shown great promise in many time-series forecasting tasks. RNNs are able to capture temporal behavior by adding a feedback connection which sends a copy of previous values of hidden states, along with the current input, as the input of a hidden cell. However, vanilla RNNs are hard to train because the vanishing and exploding gradient problem originating from repeated parameter matrix multiplication during back propagation through time. Some variations have been proposed to alleviate these problems, such as the Long Short Term Memory network [6] (LSTM) and Gated Recurrent Units [2] (GRUs), by utilizing a residual connection. The previous hidden state is added to the current cell state directly via residual connection which bypasses the parameter matrix multiplication, so LSTM and GRU can memorize longer time dependencies.

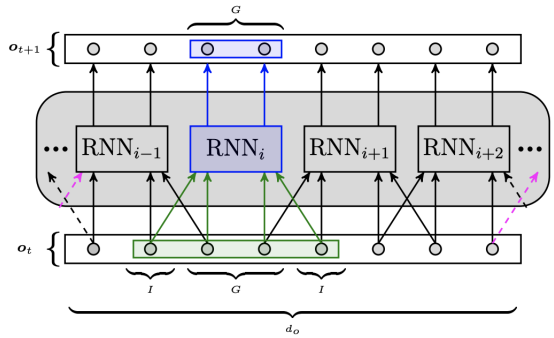

Figure 1. The parallel framework [13]. This framework trains models (..RNN$_{i-1}$,RNN$_i$,RNN$_{i+1}$..) in parallel. Each model is responsible for a spatial segment of the system G. To preserve spatial continuity of local interactions, each model accepts inputs from the spatial region it's responsible for, and from buffers to either side I.

LSTM and GRU gained significant improvement over vanilla RNNs. However, LSTM and GRU still have a sequential path from older past cells to the current one, so the length of time dependency they can learn is limited. Besides, one inevitable issue of RNN-based models is they take a long time to train due to the fact that each step depends on previous results. The proposal of Transformers [14] and BERT [3] break the dominance of RNN-based models in sequence prediction. They totally get rid of sequential order during training. The self-attention layer allows each position in the input sequence to have direct interaction with other positions with the same distance. Those attention-based models not only achieved better results on lots of Natural Language Processing (NLP) tasks than RNN-based models, but are also easier and faster to train because they can process the whole sequence at a time. In contrast to recurrent neural networks, reservoir computing (RC) does not train the recurrent connections in its network [7]. Instead, a reservoir computer initializes a random network of randomly weighted connections between hidden, non-linear units; allows input to propagate through the network; and then trains only the weights between the network and the output. In this sense, the output layer is selecting the evolution of hidden states that best match the training data [1]. This may also be thought of as a recurrent, non-linear embedding of the system used as a kernel trick. Because only one layer of weights is trained, RCs do not suffer from vanishing/exploding gradients and the training time can be very short.

Previous work has shown that reservoir computers can achieve state-of-the-art predictions for systems characterized by spatio-temporal chaos [13]. Similar results have been achieved with LSTMs, as well [16]. These techniques rely on the ability of RNNs to produce suitable non-linear trajectories for systems with extreme sensitivity to initial conditions. More recent work by a collaboration of these two groups has performed a comparative study on recurrent architectures and their ability to forecast chaos [15]. They found that RCs perform better on systems with fully observable states than other RNNs while requiring less memory and time to train. On the other hand, LSTMs and GRUs greatly outperform RCs when only partial knowledge of the system state is given. In this work, we reproduce previous experiments in comparing RNN architectures, and extend the comparison with an implementation of an attention-based model.

3. Methods

The use of sequential prediction techniques on chaotic dynamics has seen success in the past, but has been restricted to low-dimensional systems [8]. In order to predict systems with larger spatial dimension (like Lorenz's atmospheric models), these previous techniques must be augmented. To this end, we adopt the implementation of the parallel reservoir scheme proposed in [13]. This framework showed in 1 trains models in parallel — each model responsible for a spatial segment of the system. To preserve spatial continuity of local interactions, each model accepts inputs from the spatial region it's responsible for, and from buffers to either side. Thus, neighboring models will have overlapping inputs along the edges of their spatial segments (figure 1.

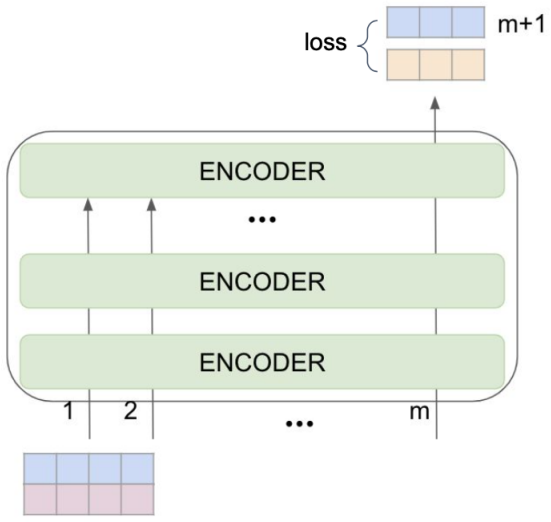

Figure 2. Our attention-based model. Like BERT, the model is a stack of Transformer encoders and each encoder is a stack of self-attention layers. The model predicts the embedding of the next step based on the previous m time steps. The input combines the space (purple) and position (blue) embeddings. On the top of the figure, the loss is calculated based on the prediction (yellow) and the ground truth (blue).

Unlike RNNs, in which the current output is fully based on previous inputs, the self-attention structure empowers each position to see the whole sequence. But for most temporal tasks in real time, it's impossible to know the future. We pick up the output of the last position to predict the embedding of the next time step after a fully connected layer. Our method only predicts one step at a time like RNN-based models, so we can't seed up the training time. Our model can gain benefit from the same distance between every two positions to model long term dependency.

Our model is depicted in figure 2. Like BERT, the model is a stack of Transformer encoders. Each Transformer encoder is a stack of self-attention layers. The model consumes embeddings of the previous m time steps and predicts the space embedding of the next $m+1$ time step. For each time step, the input embedding combines the space and position embedding. On the top of the figure, the yellow and blue bars are predicted output and ground truth respectively, the mean squared loss (which will be explained in the Experiment section) are calculated and back propagated through the whole graph. The size of the output is smaller than the input because the input incorporates some neighbour information in the parallel framework.

4. Experiment

4.1. Data Simulation

In mathematics, the Kuramoto-Sivashinsky equation (called the KS equation or flame equation) is a fourth-order nonlinear partial differential equation to model the diffusive instabilities for various phenomena. It was derived by Kuramoto [10] to model chaotic oscillatory behavior. The equation reads as:

$$\frac{\partial u}{\partial t} = -v\frac{\partial^4 u}{\partial x^4} - \frac{\partial^2 u}{\partial x^2} - u\frac{\partial u}{\partial x} \tag{1}$$

on the domain $\Omega = [0, L]$ with periodic boundary conditions $u(0,t) = u(L,t)$. In order to spatially discretize equation 1, we select a grid size $\triangle x$ with $D = L/\triangle x + 1$ as the number of nodes. Further, we select $v = 1$, $L = 200$, $\delta t = 0.25$ and a grid of $d_0 = 512$ nodes. We discretize the equation 1 and solve using the fourth-order method for stiff PDEs introduced in Kassam and Trefethen [9] up to $T = 6 \cdot 10^4$. This corresponds to $24 \cdot 10^4$ samples. The first $4 \cdot 10^4$ samples are truncated to avoid initial transients. The remaining data are divided to a training and testing dataset of 10^5 samples each.

4.2. Results on the Kuramoto Sivashinsky Equation

To assess the effectiveness of different architectures, we compared their training time to convergence, their CPU RAM memory used to train, and their average normalized root mean square error (NRMSE) (equation 2). The NRMSE is given by

$$\text{NRMSE} = \sqrt{\langle \frac{(\hat{\mathbf{o}} - \mathbf{o})^2}{\boldsymbol{\sigma}^2} \rangle} \tag{2}$$

where $\hat{\mathbf{o}}$ is the model's forecast at a time, $\mathbf{o}$ is the ground truth for the predictions, and $\boldsymbol{\sigma}$ is the standard deviation in time of each state component. The $\langle \cdot \rangle$ notation represents the state-space average of the vector.

For implementations of recurrent activation based architectures, we used the models developed in [15], and trained them on data generated by the Kuramoto-Sivashinsky equation. Some hyper-parameters were changed to be compatible with our computing environment. In particular, the number of parallel groups N_g was restricted by the number of compute nodes we had access to. For all our recurrent architectures (RC, LSTM, GRU), this parameter is set to 1. This significantly reduces the effectiveness of RC models. Where [15] reports an average NRMSE below 1.5 for all its RC models, we find values to be closer to 1.75 for RC models with a reservoir size of 1000; where they report a training time for an RC with a reservoir size of 3000 that is a small fraction of other RNNs, we find such an RC to require more time to train. The LSTM and GRU models were created with two hidden layers of 32 hidden nodes. Other parameters can be found in the appendix of [15].

When training our attention-based model in the parallel framework on the Kuramoto-Sivashinsky equation, the total number of groups N_g is set to 16. The group-size of the parallel model G is 32, as determined by d_0/N_g. The interaction length I is set to 32. The hidden size is 120 and the sequence length is 128. The number of transformer encoder layers and the number of heads in the self-attention layer are set to 4 and 2 respectively. The average NRMSE error is 1.327048 and the average running time of training is fifteen minutes. All the results are summarized in table 1.

From grid search we learned that with the same model and N_g, incorporating more neighbour information helps in decreasing the testing error. The average NRMSE error for the total number of groups $N_g \in \{4, 8, 16\}$ is 1.3834, 1.363 and 1.374 respectively. So the performance doesn't rely on the number of groups according to our experiment results.

	Training time	Memory	NRMSE
RC 1000	11s	288MB	1.771
RC 3000	85s	425MB	1.353
LSTM	81s	372MB	1.253
GRU	62s	398MB	1.253
Attention-based	15m	1G	1.324

Table 1. Results

5. Conclusion

We implement an attention-based model to predict a spatio-temporally chaotic system. The advantage of the

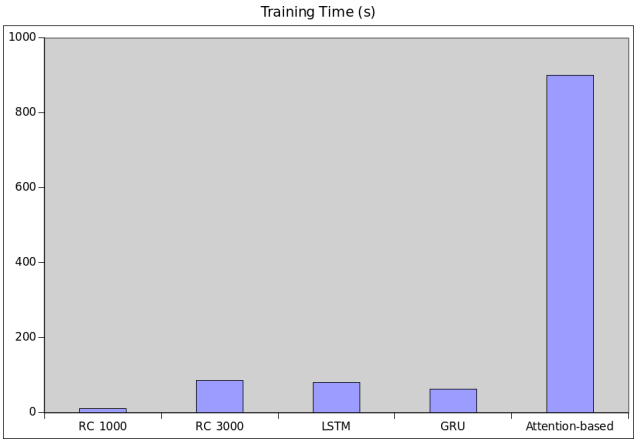

Figure 3. our attention based architecture takes an order of magnitude more time to train.

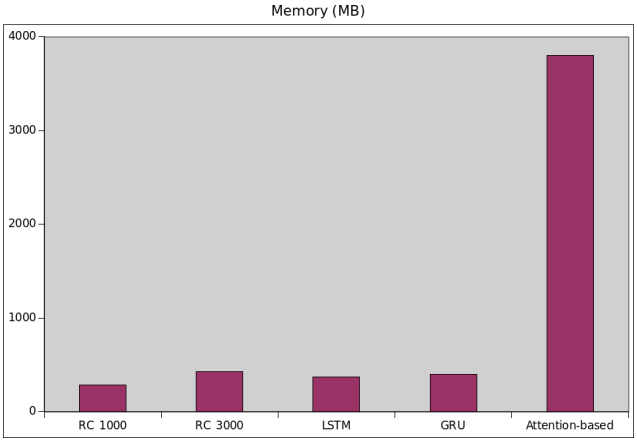

Figure 4. our attention based architecture takes an order of magnitude more memory to train.

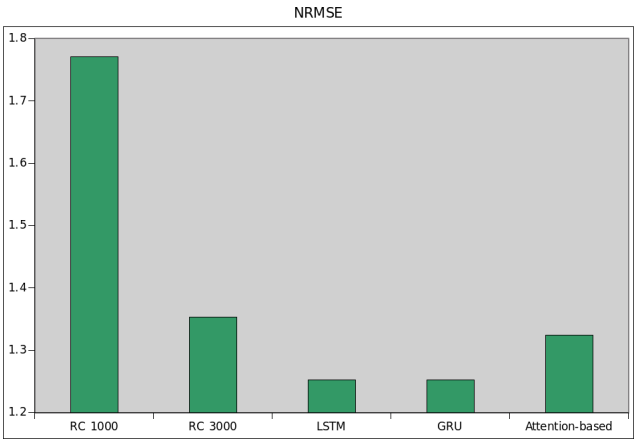

Figure 5. our attention based architecture performs about as well as recurrent architectures.

attention-based model is that every position in the sequence has the same distance from other positions, so it can capture longer-range dependencies than RNN-based models. However, like the language model in NLP, the model predicts the embedding of next step based on the previous m steps. Thus, the training process can't be sped up to process the whole sequence at a time.

In the future, we'd like to further explore the range of attention-based architectures. Theoretically, their ability to encode very long-range time-dependencies would make them ideal candidates for predicting chaotic dynamics, but our empirical results suggest otherwise. Such time-dependencies are especially important when the full dynamics of a system are unknown — a scenario where RCs perform particularly poorly. Testing attention models on a wider variety of chaotic systems could help better illuminate their strengths and weaknesses as sequence models. If it still holds that attention models are poor predictors of chaos, then that may suggest a more inherent connection between chaotic dynamics and recurrent connections.

References

[1] F. M. Bianchi, S. Scardapane, S. Løkse, and R. Jenssen. Reservoir computing approaches for representation and classification of multivariate time series. *arXiv preprint arXiv:1803.07870*, 2018. 2

[2] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 1

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[4] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 1

[5] J. Gleick. *Chaos: Making a new science*. Open Road Media, 2011. 1

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[7] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001. 2

[8] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004. 2

[9] A.-K. Kassam and L. N. Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005. 3

[10] Y. Kuramoto. Diffusion-induced chaos in reaction systems. *Progress of Theoretical Physics Supplement*, 64:346–367, 1978. 3

[11] M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014. 1

[12] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963. 1

[13] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.*, 120:024102, Jan 2018. 1, 2

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1, 2

[15] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. 2019. 1, 2, 3

[16] Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5), 2018. 2