The COMP6258 Reproducibility Challenge
Encoding Recurrence into Transformers

Rory Coulson, 31637078, rc3g20, Yi Ren Chang, 32440855, yrc1g20, William Raftery, 31680615, wr1g20, Dávid Máté, 30756731, dm1e19
https://github.com/COMP6258-Reproducibility-Challenge/COMP6258-Encoding-Recurrence-into-Transformers

1 Introduction

This report aims to reimplement the time series forecasting and natural language modeling experiments with the proposed recurrence encoded Transformers presented in the ICLR 2023 paper "Encoding Recurrence into Transformers" [2], and determine the reproducibility of the results.

1.1 Background

Transformers have revolutionized the field of natural language processing due to their parallelization and improved learning of long-range dependencies, typically achieving superior performance to alternative models [3]. Transformers have also demonstrated significant potential in sequential learning tasks, such as time series forecasting [4], due to these long-range dependencies. However, the flexibility of Transformers may result in inefficient sampling to achieve good generalization, typically discarding the chronological ordering of the data. Alternatively, recurrent models such as LSTMs and GRUs have demonstrated notable success when applied to sequential learning tasks, primarily due to their recurrent inductive bias. Yet, recurrent models often struggle to establish strong correlations between distant inputs due to the vanishing gradient problem, despite efforts to mitigate this through the use of long memory patterns. Recurrent models are also known to be challenging to train in parallel, resulting in longer training times [2].

1.2 Methodology

The authors propose a novel approach to encoding recurrence into the Transformer architecture, aiming to retain the strengths of both models while avoiding their weaknesses. This approach introduces recurrent dynamics of an RNN into the position encodings of a Transformer via a 'Recurrence Encoding Matrix (REM)', as depicted in Figure 1d, to boost the sample efficiency of the Transformer, leading to a new module, 'Self-Attention with Recurrence (RSA)' [2], as shown in Figure 1a-c.

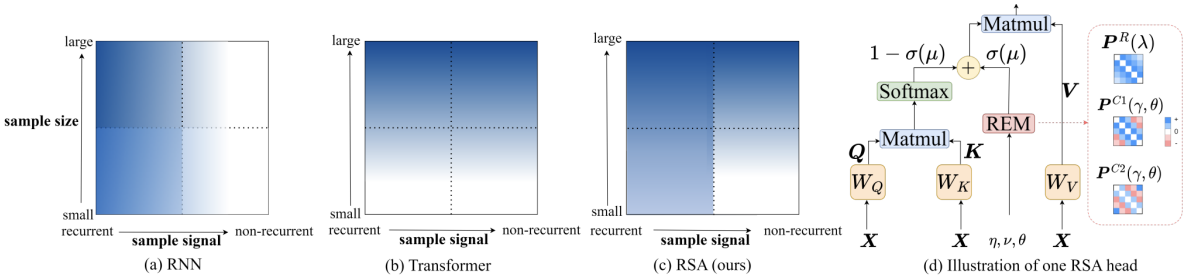

Figure 1: Comparative sample efficiency of the RSA and the RSA head structure [2]

The RSA module can additionally be run in parallel, with authors displaying how an RNN layer with linear activations can be split into multiple simple RNNs with 'scalar hidden coefficients,' which collectively form the recurrent dynamics [2]. Moreover, this can be written into the positional encodings of a multi-head self-attention mechanism.

2 Implementation

The original paper presents the 'Self-Attention with Recurrence (RSA)' module that updates the baseline transformer's self-attention mechanism, shown in Equation 1 [2].

$$RSA(X) = \{[1 - \sigma(\mu)]softmax(QK') + \sigma(\mu)P\}V \tag{1}$$

The learnable parameter, μ, of the RSA module is defined for all attention layers and $\sigma(\mu)$ acts as a gate to control the modelling ratio of recurrent and non-recurrent signals. P represents the REM module, with six types of REMs, one regular, two cyclical and their respective dilated versions, to be chosen for each head, controlled by hyperparameters k_i with dilated factors d given in the paper. The REM forward method takes in the initialized parameters (η, ν, θ) and computes the REM via:

$$P = \begin{cases} \lambda^L - I_T & \text{for regular REM} \\ \gamma^L \odot \cos(\theta \odot L) & \text{for first cyclical REM} \\ \gamma^L \odot \sin(\theta \odot L) & \text{for second cyclical REM} \end{cases} \tag{2}$$

where L is a (TxT) Toeplitz matrix, unless truncated. To avoid explosive results the parameters $\lambda = \tanh(\eta)$ and $\gamma = \sigma(\nu)$ are further bounded. The REMs are masked, converted to lower triangular matrices, or unmasked depending on the given task, as specified in the original paper [2]. The dilated REMs can be computed by $P \bigotimes I_d$, given the dilated factor d.

A main challenge in this reimplementation originated from the possible inconsistencies found between the given guidelines, pseudo code and the paper's detailed methodology [2]. For example, the pseudo code suggests to have regular REMs of shape (n_heads x query_len x key_len) concatenated with cyclic REMs of the same size, leading to double the expected heads in the later summation, shown in Equation 1. Additionally, the REM pseudo code suggests only k_{2-4} are necessary for its development, contrasting with the original paper. This could be due to the simplified nature of the pseudo code, however, significant alterations were made to best align with the paper's original methodology.

3 Results

3.1 Informer and RSA-Informer Reproducibility

The first experiment conducted was investigating the reproducibility of the baseline Informer[1] and modified RSA-Informer time series forecasting results applied to the ETT (Electricity Transformer Temperature) and weather datasets presented in the original Informer paper [4]. The results are given in Table 1 with the (R) tag identifying the reproduced findings. The bold values indicate where the RSA-Informer out-performed the baseline Informer, using the Mean Squared Error (MSE) and Mean Absolute Error (MAE) as metrics.

Overall, our findings display mixed results for the benefits of the RSA-Informer compared with the results from the original paper. Most notably, experiments conducted on the ETTh1 dataset demonstrated the most similar pattern between the two models, while the other datasets generally display poor correlation and possibly no benefit to the baseline Transformers.

The first issue experienced was the mismatching shown between the baseline Informer results. These results are expected to differ from the original Informer paper [4], due to the change from random to sequential sampling of the time series data. However, our attempt to apply this sampling by the removal of random shuffling in the Informer still displays significantly differing results, shown in Table 1. This likely indicates additional changes are required but implementation details from the paper were found to be lacking.

Further issues were found replicating the RSA-Informer pattern of improved performance against the baseline. This could be due to the truncation of the REM module that had to be applied in order to match the non-square QK'

[1] Available at https://github.com/zhouhaoyi/Informer2020

Methods		Informer		RSA-Informer		(R) Informer		(R) RSA-Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	24	0.762	0.632	**0.414**	**0.450**	0.679	0.590	**0.632**	**0.568**
	48	1.006	0.763	**0.467**	**0.493**	0.935	0.724	**0.908**	**0.714**
	168	1.141	0.823	**0.753**	**0.659**	1.464	1.000	**1.265**	**0.886**
	336	1.416	0.987	**0.895**	**0.755**	**1.645**	**1.071**	1.902	1.158
ETTh2	24	2.558	1.253	**1.264**	**0.879**	**0.951**	**0.777**	1.091	0.840
	48	2.487	1.268	**1.878**	**1.067**	2.584	1.300	**1.465**	**0.970**
	168	2.869	1.324	**2.830**	**1.301**	**2.534**	**1.259**	3.658	1.559
	336	**2.055**	**1.113**	2.113	1.124	**2.292**	**1.195**	2.331	1.222
ETTm1	24	0.536	0.511	**0.534**	**0.507**	**0.695**	**0.6114**	0.7411	0.643
	48	0.781	0.633	**0.644**	**0.612**	0.782	**0.648**	**0.747**	0.744
	96	0.823	0.697	**0.732**	**0.665**	**1.088**	**0.817**	1.313	0.935
	288	1.371	0.945	**0.835**	**0.710**	X	X	X	X
Weather	24	**0.316**	**0.371**	0.328	0.380	**0.886**	**0.720**	0.992	0.765
	48	0.606	0.566	**0.432**	**0.464**	0.555	0.546	**0.490**	**0.502**
	168	1.009	0.771	**0.862**	**0.702**	**0.643**	**0.598**	0.853	0.699
	336	1.096	0.801	**0.846**	**0.697**	X	X	X	X

Table 1: Informer and RSA-Informer Reproduced Results for Time Series Forecasting

matrix from Equation 1, since the Informer model selects only the top-u queries based on their sparsity [4]. We found the paper does detail REM modification for non-square attention shapes but addresses this from an overlapping period case, causing difficulty in interpreting the correct truncation function for the Informer. Lastly, the missing results seen for ETTm1 and weather datasets exist where our single GTX 1080 GPU exited due to memory constraints.

3.2 Transformer-XL, RSA-XL and BRT Reproduciblility

The Transformer-XL [1] architecture was used to evaluate the impact of the RSA module on language modelling tasks. Baseline results were generated using code from the official Transformer-XL repository[2] which was then modified to create the RSA-XL.

The hyperparameters were set as specified in the paper. Those hyperparameters which are omitted from the paper were set to the values used in the original Transformer-XL implementation. We also tested a smaller version of the RSA models to decrease training times. The number of layers was decreased from 14 to 7, and the model dimension was decreased from 512 to 8. The full list of parameters can be found in the scripts 'run_enwik8_rsa_small.sh' and 'run_text8_rsa_small.sh' of the repo.

However, the training times were unfeasible with the computational resources available (roughly 37 hours on 4 GTX 1080Ti GPUs). We therefore lowered the maximum number of training steps to 5% of the original value (from 400 000 to 20 000) for the baseline and RSA-XL-Small models, and down to 10 000 steps for the RSA-XL model. Consequently, it is expected that the results will differ from those given in the paper. However, we can still identify whether the same trend is present in the results. The results are shown in Table 2.

Models	XL	RSA-XL	RSA-XL-Small	BRT
Enwik8	1.26	5.07	3.93	1.29
Text8	1.28	4.10	3.19	1.30

Table 2: Natural Language Modelling: Bit-per-character (BPC)

The RSA models did show a downward trend in the validation loss over epochs. However, this trend was very slow and the RSA models performed worse than the baseline after 20 000 training steps. The baseline models had a sharp decrease in validation loss over the first 300-800 training steps. The rate of change quickly slowed to a much lower rate, and then remained steady for the rest of training. The RSA models did not exhibit this initial drop. The rate of

[2]Availible at https://github.com/kimiyoung/transformer-xl

decrease in the training loss remained low throughout training. The RSA-XL-Small model showed a better learning dynamic during training, with the validation loss decreasing significantly over the 20 000 training steps.

There were a number of challenges in implementing this model, including a lack of detailed requirements and the use of an outdated version of PyTorch.

Table 7 in Appendix E.3 of the paper specifies exactly which hyperparameters were used. The table also includes the number of parameters for each model. Our parameter count aligned with these values for the baseline models. However, our RSA-XL consistently had 14 fewer parameters than specified. Our RSA-XL models had 56 more parameters than their baseline versions. This is due to the four trainable parameters (μ, ν, η, θ) in the RSA module times the 14 layers. This implies that we have overlooked a learnable parameter despite thoroughly examining the paper, which could be responsible for our models poor performance.

The BRT model was trained using the training schedule of the Transformer-XL implementation, as specified in the paper. However, the number of tokens and the inner dimension were not given, which made reproducing the model with the same parameter count difficult. Due to time limitations, the network was trained for 10 % of the maximum number of iterations, which explains why the BPC is lower than the original.

4 Conclusion

Overall, our efforts to reproduce the experiments described in the original paper have yielded mixed results. We successfully replicated some aspects of the experiments, particularly those related to the ETTh1 dataset, where the RSA-Informer model showed a performance pattern similar to that reported in the original paper. However, the Enwik8 and Text8 datasets did not show the same level of correlation, suggesting potential issues with the reproducibility of the results.

Several challenges impeded our attempts at replication. A significant issue was the inconsistencies between the provided guidelines, pseudo code, and the detailed methodology described in the paper. For example, the discrepancy between the regular REMs' shape in the pseudo code and the paper's description led to confusion and required us to make substantial alterations to align with the paper's methodology. Additionally, the lack of detailed implementation steps for modifications required by non-square attention shapes posed further difficulties.

Resource constraints also played a crucial role in our findings. The computational resources required for training, particularly for the RSA-XL model, were substantial. Our available hardware, 4 GTX 1080 GPU, was insufficient to fully replicate the experiments, leading to incomplete results for certain datasets due to memory limitations. We had to reduce the number of training steps to get the results.

Despite these challenges, the paper does a commendable job in proposing a novel approach to integrating recurrent dynamics into Transformers. However, additional details on specific implementation steps and clearer alignment between the pseudo code and methodology would likely have improved the reproducibility of our findings. Enhanced documentation and updated dependencies in the official repositories would also aid future researchers in replicating these experiments more effectively.

References

[1] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019.

[2] Feiqing Huang, Kexin Lu, CAI Yuxi, Zhen Qin, Yanwen Fang, Guangjian Tian, and Guodong Li. Encoding recurrence into transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[4] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.