CEMRACS 2023 project proposal: Structure-Preserving Numerical Integration of Dynamical Systems using Transformer Neural Networks

1 Context

Two of the many trends in neural network research of the past few years have been (i) the "learning" of dynamical systems, especially with recurrent neural networks such as LSTMs [9, 2, 10], and (ii) the introduction of transformer neural networks for natural language processing (NLP) tasks [8]. Both of these trends have created enormous amounts of traction, particularly the second one: transformer networks now dominate the field of NLP.

The reasons for this are essentially two-fold: one is the simplicity and interpretability of the transformer architecture and the other one is its parallelizability for computation on GPUs. While vanilla recurrent neural networks also lend themselves to almost straightforward interpretation, LSTMs (their more succesful version) do not, as the network architecture involves many complex components. In addition, all recursive neural networks are sequential by design, which makes optimization on a GPU impractical.

Some efforts have already been made to include elements of the transformer architecture into neural networks that are designed to learn dynamical systems [7, 1], but although showing some success, these approches only utilize part of the transformer network. This negligence means that the success that transformers have experienced in NLP has not been extended to dynamical systems.

2 Description and objectives

The original transformer paper [8] proposed a relatively simple neural network architecture that is easy to train (allows for straightforward parallelization) and shows remarkable success when applied to NLP tasks.

Even though the transformer contains many components (for example vanilla feedforward neural networks), the two core components are a "positional encoding" and a "attention mechanism"; the second component appears several times in the transformer. The positional encoding is a mapping that takes a set of vectors $\xi_1, \dots, \xi_T$ (i.e. time-series data; in the original paper ξ_i represents the i-th word in a sentence of length T) and produces a new set of vectors $\tilde{\xi}_1, \dots, \tilde{\xi}_T$ that "are aware" of their position in the sentence/time series. For the original transformer paper this was done in the following way:

$$\xi_i^j \mapsto \tilde{\xi}_i^j := \xi_i^j + \begin{cases} \sin(i \cdot 10^{-4j/e}) & \text{if } j \text{ is even} \\ \cos(i \cdot 10^{-4(j-1)/e}) & \text{if } j \text{ is odd,} \end{cases} \tag{1}$$

where e is the length of the vector ξ_i, i.e. the dimension of the "embedding space". In mathematical terms this positional encoding adds an element of the torus $\mathbb{T}^{\lfloor e/2 \rfloor}$ to the input ($\lfloor \cdot \rfloor$ is the floor or "rounding down" operation). Even more concretely, the embedding corresponds to the solution of a system of $\lfloor e/2 \rfloor$ independent harmonic oscillators with frequencies $10^{-2j/e}$ (for $j \in \{1, \dots, \lfloor e/2 \rfloor\}$); $i \in \{1, \dots, T\}$ in the above equation is the time step. The vastly different frequencies of the harmonic oscillators (very high for low j and very low for high j) are meant to capture different features of the data.

The other core component is an "attention mechanism" whose inputs are sets of "queries" Q, "keys" K and "values" V (in the original paper each row of these matrices represents a word in a sentence) and performs a reweighting of $\hat{V} := VW^V$ based on $\hat{Q} := QW^Q$ and $\hat{K} := KW^K$:

$$\text{Attention}(\hat{Q}, \hat{K}, \hat{V}) = \text{softmax}\left(\frac{\hat{Q}\hat{K}^T}{\sqrt{e}}\right)\hat{V}. \tag{2}$$

This realizes a position-dependent reweighting of the each row in $\hat{V}$ (each row of V or $\hat{V}$ represents a word, previously called ξ_i); the matrices W^Q, W^K and W^V are "projection matrices" that are learned during training. For NLP tasks this reweighting step is crucial as it assigns higher value to words that contain meaning and lesser value to words such as arcticles ("a", "the") that do not.

The objective of this project is the adaptation of the transformer architecture, especially the two crucial components (the positional encoding and the attention mechanism) to dynamical systems. A point of interest will be investigating the role that the positional encoding plays for such systems; in

the case of the original transformer this adds a simple dynamical system to the input - this may be superfluous if the input is already a dynamical system.

An effort should also be made towards making the network architecture "structure-preserving" (i.e. such that the resulting integration scheme is symplectic). A possible way to do this is to design the new networks in a similar way to SympNets [4], i.e. updating the q and p components of a canonical Hamiltonian system separately.

3 Proposed methodology

The project can roughly be divided into four steps.

In the initial stage of the project some familiarity with the transformer architecture should be gained; this may involve implementing the original transformer [8].

The next stage will be adjusting the attention mechanism (equation (2)) to dynamical systems: this will perhaps require changing the activation function from a softmax to something different.

A third step will be investigating different choices of input embeddings to the original one (equation (1)). As this is essentially an addition of the input by the solution of a simple dynamical system, it should be possible to improve on this.

Lastly, efforts should be made to make the resulting network structure-preserving, i.e. symplectic, in a similar way as was done for SympNets.

4 Software Requirements

The project will be implemented in Julia. Aspects of the Lux [6] and Flux [3] libraries will be used for the machine learning part and the library GeometricIntegrators [5] will be used for generating training data. The results will be integrated into the library GeometricMachineLearning.

References

[1] Nicholas Geneva and Nicholas Zabaras. "Transformers for modeling physical systems". In: *Neural Networks* 146 (2022), pp. 272–289.

[2] Jesús Gonzalez and Wen Yu. "Non-linear system modeling using LSTM neural networks". In: *IFAC-PapersOnLine* 51.13 (2018), pp. 485–489.

[3] Michael Innes et al. "Fashionable Modelling with Flux". In: *CoRR* abs/1811.01457 (2018). arXiv: `1811.01457`. URL: `https://arxiv.org/abs/1811.01457`.

[4] Pengzhan Jin et al. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems". In: *Neural Networks* 132 (2020), pp. 166–179.

[5] Michael Kraus. *GeometricIntegrators.jl: Geometric Numerical Integration in Julia.* `https://github.com/JuliaGNI/GeometricIntegrators.jl`. 2020. DOI: `10.5281/zenodo.3648325`.

[6] Avik Pal. *Lux: Explicit Parameterization of Deep Neural Networks in Julia.* `https://github.com/avik-pal/Lux.jl/`. 2022.

[7] Anna Shalova and Ivan Oseledets. "Tensorized transformer for dynamical systems modeling". In: *arXiv preprint arXiv:2006.03445* (2020).

[8] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[9] Yu Wang. "A new concept using lstm neural networks for dynamic system identification". In: *2017 American control conference (ACC)*. IEEE. 2017, pp. 5324–5329.

[10] Yanwen Xue, Jun Jiang, and Ling Hong. "A LSTM based prediction model for nonlinear dynamical systems with chaotic itinerancy". In: *International Journal of Dynamics and Control* 8 (2020), pp. 1117–1128.