
Transformers for modelling Physical systems

Anuleela kethe
am1200640@am.iitd.ac.in

Abstract

Transformers are widely used in the Natural language processing due to their ability for long term dependencies. These models are a type of deep learning model mostly used in natural language processing (NLP) tasks. These have little applicability outside of the natural language processing. In this work, we use transformers for predicting the dynamical representation of the physical phenomenon. Transformers rely on the self-Attention that allows the transformer to learn temporal dependencies by using Koopman embedding technique which is used for all dynamical systems to project any dynamical system into a vector representation.

1 Introduction

Transformer models built on self-Attention for large tasks related to natural language processing. Transformers are becoming state of art approach for many data sets. In the deep learning models there are many models such as Auto-regressive, Recurrent Neural Network, Long short Term Method (LSTM) that are also used to model the physical systems. In the recent times, transformers are entirely relying on the Self-Attention to model dynamics. In this work, a physics inspired embedding model is used to model the physical systems.

2 Methods

Mainly we focus on the dynamical systems which are in the form of differential equations or partial differential equations:

$$\begin{aligned}\phi_t &= F(x, \phi(t, x, \eta), \nabla_x \phi, \nabla_x^2 \phi, \phi \cdot \nabla_x \phi, \dots), \quad F: \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n, \\ t &\in \mathcal{T} \subset \mathbb{R}^+, \quad x \in \Omega \subset \mathbb{R}^m,\end{aligned}$$

in which $\phi \in \mathbb{R}^n$ is the solution of this differential equation with parameters η , in the time interval \mathcal{T} and spatial domain Ω with a boundary $\Gamma \subset \Omega$. In the problem, the continuous solution is discretized into both spatial and temporal domains. The discretized solution $\phi = \{\phi_0, \phi_1, \dots, \phi_T\}$, $\phi_i \in \mathbb{R}^{n \times d}$ for which ϕ_i has been discretized by d points. The dynamical problem is posed as a time-series problem.

There are two core components:

1. Embedding for projecting physical state of dynamical problem into the vector representation
2. Training the transformer input as embedding

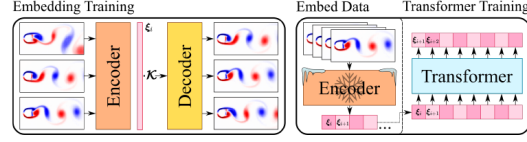


Figure 1: The two training stages for modelling the physical systems using transformers.

In the above figure 1 , we can see the two stages, first we train the embedding model with Koopman operator and then embedded training data is trained using the transformer followed by conversion of the embedding space into the physical state using the embedding decoder.

2.1 Transformer

Originally Transformers are designed with the NLP with word2vec embedding . Embedding model is used for physical systems whereas Self-Attention is used for the time-series problem. Input to the Transformer is embedding space $\mathbb{E} = \{\xi_0, \xi_1, \dots, \xi_T\}$, where the embedded state at time-step i is denoted as $\xi_i \in \mathbb{R}^2$.

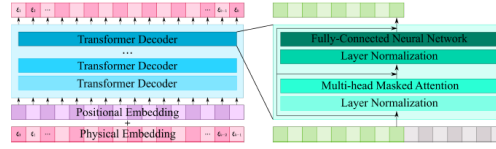


Figure 2: Transformer decoder used for predicting the dynamical system

We use the Transformer model 2 which is also used in the Generative pretrained models (GPT) models

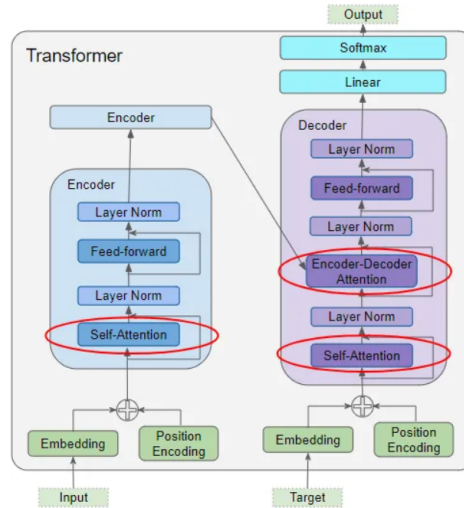


Figure 3: Transformer model used in the NLP

Main components of Transformer :

2.1.1 Word Embedding

The transformer has two embeddings, one in the encoder and another in decoder. We generally pass the test sequence IDs 4 which is mapped as text sequence to IDS according to the word vocabulary to embeddings.

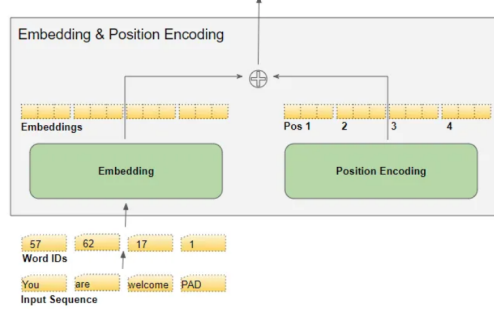


Figure 4: Embedding

2.1.2 Positional Encoding

RNN implements the for loop where it knows the position of the input explicitly. Since transformers take input inparallel, information regarding position is lost. Therefore we incorporate the positional encoding to the achieve the positional information both at the encoder and decoder stacks which have the same dimensionality d_{model} as the embeddings.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

where pos is position of the word in the sequence, d_{model} is the length of the encoding vector, i is the index of the vector

2.1.3 Self-attention

In context of NLP, in this component we find how the word is related to other words using Query (Q), Key (K) and value (V) matrix. The output is the weighted sum of the values of the input where weight is the self-Attention function, and Q, K and V are calculated by multiplying embedded word vector with W_q , W_k and W_v matrices which are of size (d_{model}, d_{model}) . Whereas in the context of physics, it is used to learn the temporal dependencies. However, there are many attention models, commonly used is scaled-dot product Attention, which are more faster and more space-efficient in practice.

Scaled-dot product :

$$k_i = \mathcal{F}_k(x_i), \quad q_i = \mathcal{F}_q(x_i), \quad v_i = \mathcal{F}_v(x_i)$$

$$c_n = \sum_{i=1}^k \alpha_{n,i} v_i, \quad \alpha_{n,i} = \frac{\exp(q_n^T k_i / \sqrt{d_k})}{\sum_{j=1}^k \exp(q_n^T k_j / \sqrt{d_k})},$$

where k, q, v are the keys, queries, values which are learned through the neural networks, c_n is output, $\alpha_{n,i}$ is the Attention score. The attention scores always sum to one, $\sum_{i=1}^k \alpha_{n,i} = 1$, for every input.

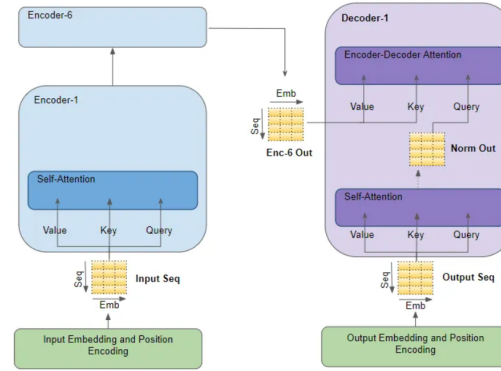


Figure 5: Three different self-Attention

Attention used at Three positions in Transformer:

1. **Attention used in encoder:** The input embedding and the positional embedding which has positional information for each word. Through the self-Attention, the Attention score is calculated for each word, then it passes to next stack
2. **Attention used in decoder:** The target is passed to output embedding and positional embedding, it passed to the self-Attention, attention score is calculated here then passed to the norm layer where it is fed to query parameter.
3. **Encoder-Decoder-attention in the Decoder:** At the encoder-Decoder Attention, the output from the encoder stack is passed to the key and value parameters and the attention score is added at each self-attention.

2.1.4 Multi-Head attention

All the Attention modules run in parallel, outputs from each attention heads are concatenated to produce the final Attention score. This gives greater power for transformer to encode multiple relationships for each word.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

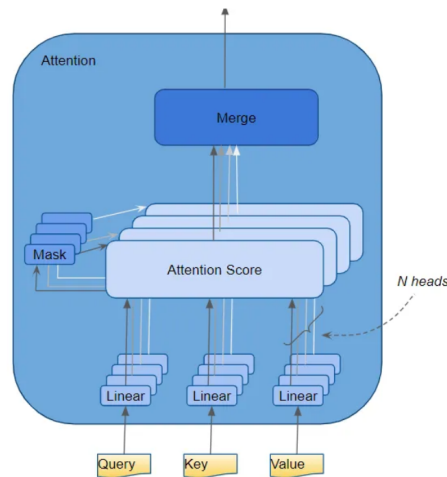


Figure 6: mutli-head attention

2.2 Embedding Model

2.2.1 Koopman Operator Theory

Koopman Provided a alternative for the representation of the dynamical systems in terms of evolution functions. The Koopman operator \mathcal{K} is an infinitesimal dimensional linear operator which convert the non-linear dynamical systems into the linear for predicting and estimation of the comprehensive theory.

$$\mathcal{K}g \triangleq g \circ F \Rightarrow \mathcal{K}g(x_k) = g(x_{k+1})$$

2.2.2 Embedding model

In the NLP, after tokenization of the words, then they are embed using various models such as word2vec etc. For the dynamical systems we use the koopman embedding model which is universal method. We use the approximate koopman observations as a methodology. The koopman embedding is trained with the following loss function:

$$\mathcal{L}_{D_k} = \sum_{i=1}^D \sum_{j=0}^T \underbrace{\lambda_0 \text{MSE}(\phi_j^i, G \circ F(\phi_j^i))}_{\text{Reconstruction}} + \underbrace{\lambda_1 \text{MSE}(\phi_j^i, G \circ \mathcal{K}^j F(\phi_0^i))}_{\text{Dynamics}} + \underbrace{\lambda_2 \|\mathcal{K}\|_2^2}_{\text{Decay}}$$

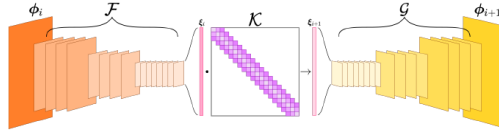


Figure 7: Koopman Embedding model for 2D system using convolution encoder-decoder model

The Koopman observable embedding has the similar motivation as of word2vec. The main objective is not finding the Koopman observables but to leverage koopman to get the physical context.

3 Chaotic Dynamics

Here we model the Lorenz system, well known for chaotic dynamics which is more sensitive to the numerical perturbations, is governed by the following equations:

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z$$

The proposed transformer and embedding model are trained for 200-300 epochs. Here the embedding model is a fully-connected encoder-decoder.

In the figure 8, the proposed transformer model and the alternative models such as fully-connected auto-regressive model, a fully-connected LSTM model, a deep neural network Koopman model and an echo-state model relative mean is shown. We can see that the transformer model worked exceptionally with less number of parameters.

5 Conclusion

Transformers and self-Attention are performed well in NLP. In this we discussed the transformers for dynamical systems. The main challenge of this transformer model is the find the appropriate representation of the physical state of system so we introduced the koopman embedding model. Transformer models outperformed than other models like fully-connected auto-regressive model, a fully-connected LSTM model, a deep neural network Koopman model and an echo-state model.

As of update, initially started with understanding the code and logic, but unable to run on given examples. This has created lots of work and ended up in wasting lots of time solely due to lack of compatibility. Code was written in PyTorch and was supposed to be converted into Tensorflow but original code was not working well with example. This has changed the original motive of assignment to change making it run which ultimately was not attained.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.

[2] <https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a6>

[3] <https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1>

[4] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. Nature communications, 9(1):1–10, 2018.

[5] <https://github.com/zabaras/transformer-physx>