# Final Project Report

# Applying Time Series Forecasting Model to Decision Transformer

Zhaolin Qiu
Binghamton University
EECE568-01

**ABSTRACT**

Decision Transformer applied the transformer architecture and self-attention to the filed of offline RL. Despite the several advantages transformer has, its complexity and high hardware requirements limit implementation for RL tasks in real world. This project applied the modifications of self-attention that were developed or widely used for time series forecasting to the Decision Transformer architecture since both solve problems by making predictions. Two models, Informer and Reformer were adapted (made causal) and replaced the GPT model in the original Decision Transformer. All models are trained using D4RL data sets in Gym environment. The Experiments showed that the modifications did not result in a significant loss of performance. This project also introduced a noise test to further test the stability of such architectures under non-ideal scenarios. With a 5% probability, the actions were changed from original data sets. The model was then trained on the altered actions. The result shows that the self-attention mechanism is able to maintain the same level of performance under noisy observations. The experiments imply a similar utilization of self-attention mechanism between the forecasting and Decision Transformer. Modifications and improvements developed for forecasting task may also be adapted to the Decision Transformer to reduce its complexity while keep the strength of self-attention mechanism.

## INTRODUCTION

Transformer uses self-attention mechanism to extract and focus on key information from complex data.[2] Although originally developed for natural language processing(NLP) tasks, transformer was soon adapted to other fields such as computer vision(CV), times series forecasting and achieved or even outperformed the State-Of-The-Art algorithms. Decision Transformer model introduced the self attention for RL tasks via sequence modeling and supervised learning method.[1]

However, the strength of self-attention also sets a higher requirement for hardware resources. Variants of transformers are developed to reduce its time and memory complexity.[] Since both time series forecasting model and the decision transformer make predictions, the modifications of transformer for forecasting tasks may be applicable to simply the decision transformer model for RL tasks.

This project will adapt the modifications of self-attention made for forecasting tasks to the decision transformer model and evaluate the performance of the new models. The project performed a noise test to explore the potentials of such architecture implemented in non-ideal scenarios for real world problems.

## BACKGROUND

### Offline RL

In typical online RL algorithms, the agent interacts with the environment and observe the states. The learning process is usually based on value functions as the goal of the agent is to get maximum expected returns. In real world, online RL approaches are limited by many constraints. For example, when interactions between the agent and the environment are dangerous or expensive, online RL approaches are not preferred.
Offline RL is introduced to avoid direct interactions between the agent and the environment. Instead of the environment access, the agent only has access to a pre-collected set of trajectories.

### Self-attention and Transformer

Self-attention is the fundamental component in the Transformer model. The attention matrix compute a weighted sum of all input tokens for each output token. Each weight indicates the relevance (attention) of an output element to an input element. This mechanism allows the model to focus on different parts of the input sequence.[2]
The transformer contains two sections: encoder and decoder. A GPT model which the decision transformer uses, only utilized the decoder section.[10] In general, transformer is a block based architecture, a big transformer model can be easily developed by stacking the blocks to handle complex and long input sequence.

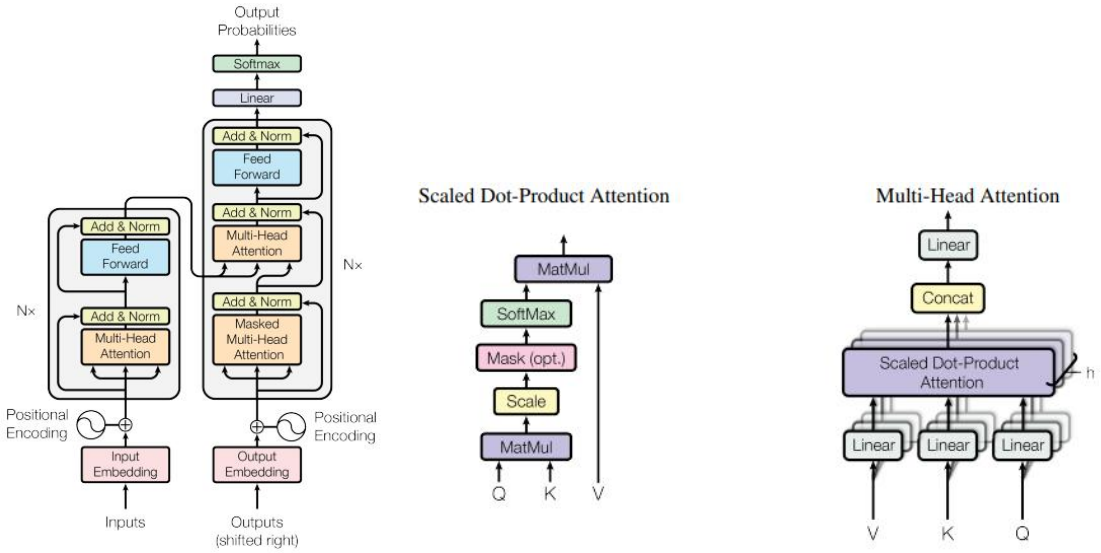


Figure 1: Transformer Architecture and Self-attention module

Despite the strength of self-attention, the quadratic time and memory complexity limits the implementation of transformer. Many variants of transformer were developed to overcome this weakness.[4]

**Decision Transformer**

Decision Transformer embeds the states, actions and returns into a sequence. Then a causal Transformer (GPT, BERT, etc) is applied to predict the next action. The input sequence has a length K time steps and can be defined as:

$$\tau = (R_{t-K+1}, s_{t-K+1}, a_{a-K+1}, ..., a_{t-1}, R_t, s_t)$$

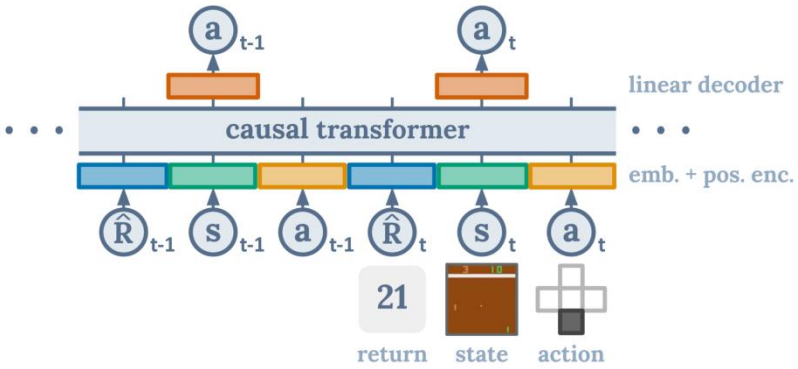


Figure 2: Decision Transformer Architecture

Decision Transformer predicts the next action based on the input sequence. During the training process, the model reduces the difference between the predicted actions and the real actions in the data sets, which can be described as:

$$\Pr(a_t \mid \tau) = \Pr(a_t \mid R_{t-K+1}, s_{t-K+1}, a_{a-K+1}, ..., a_{t-1}, R_t, s_t)$$

Unlike other supervised learning based algorithms such as behavior cloning (BC), the transformer model is able to utilize the non-optimal trajectories while BC typically discards

these trajectories since they will decrease the performance.[4]
The self-attention gives the transformer the ability of generalization. This becomes more obvious as the model gets larger. A Google Team showed that the transformer model that was trained on one atari game was able to achieve near-human performance on other similar atari games which the model was never trained on.[7]

## APPROACH

Decision Transformer Model makes decisions by making predictions on actions. The Transformer has been applied in forecasting task which makes predictions on the future given the past information. Both types of models are sensitive to sequence length. As the sequence length increases, the performance increases dramatically.[1] While long sequences are desired for better performance, it put a heavy load on the hardware. Facing the same need, this project attempts to apply the improvement found to the Decision Transformer.

Many State-Of-The-Art forecasting model modify the self-attention module. In the field of NLP, the full attention is well utilized. More efficient transformer variants are generally less powerful than the original transformer in terms of performance[4]. For prediction tasks in the filed of time series forecasting, modifications to the self-attention module generally won't lead to a significant loss of performance.

This project will applied two models that are widely used in the flied of forecasting: Reformer[8] and Informer[9].

### Reformer

Reformer manged reduce the complexity from O(L^2) to O(LlogL) by replacing the dot-product attention module with locality-sensitive hashing and using reversible residual layers. A simpler attention module lowers the hardware requirements and allows an implementation of longer sequence.

### Informer

Unlike Reformer which was developed for a general use but later applied to forecasting tasks due to its strength in long sequence predictions, Informer was developed primarily for extremely long sequence tasks. Informer proposed a ProbSparse self-attention mechanism

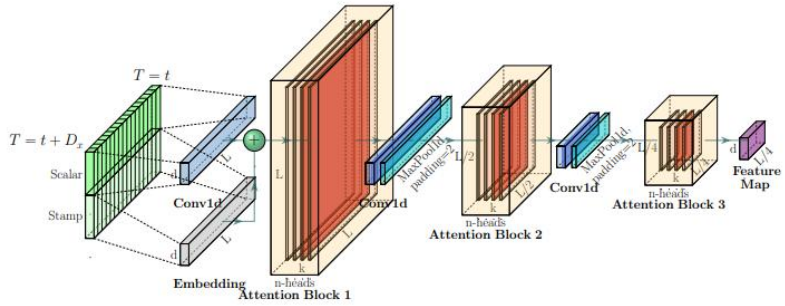


Figure 3: ProbSparse self-attention in Informer

## EXPERIMENTS

### Data Sets and Environments

This project uses D4RL as the data sets. As a popular benchmark for offline RL algorithms, D4RL provides various types of environments with trajectories generated using different policies. The data set also provides standard evaluation scores which makes the evaluation and comparison much more convenient.[3]
Due to limited time and resources, all models were only trained and evaluated on the following three environments: Walker-Medium, Hopper Medium and HalfCheeta-Medium.

### Replacement Experiment

In this experiment, the GPT model in the Decision Transformer were replaced by Reformer and Informer. However, both models can not be directly applied. They have to be modified such that they are causal and auto-regressive.

For Reformer, only the encoder part is sufficient and the parameter causal was set to true. For informer, the model itself does not give the option to be causal, a larger modification is needed. Through experimenting, I remove the encoder, the cross-attention in the decoder section and final projection layer of the decoder. The left parts of Decision Transformer were left unchanged. Both models use the same hyper-parameters as the original paper.
The results are shown in Table 1.

Table 1: Evaluation of Decision Transformer, Reformer, and Informer in Gym Environment using D4RL data sets

| Dataset | **Replaced by Reformer** | **Replaced by Informer** | DT (official) | CQL (baseline) | BC (baseline) |
|---|---|---|---|---|---|
| Walker-Medium | 77.8 | 71.3 | 74.0 | 79.2 | 77.3 |
| Hopper-Medium | 59.1 | 64.3 | 67.6 | 58 | 63.9 |
| HalfCheeta-Medium | 42.0 | 42.2 | 42.6 | 44.4 | 43.1 |

Note that due to the limitation of computing resources, the new models were trained for 20k-40k iterations while the official Decision Transformer was trained on 100k iteration. Therefore my results have greater variance than the official results.

### Action Noise Experiment

Although Offline RL avoid direct interactions between the agent and the environment, it is still challenging to collect desirable data sets. A data set of good quality usually does not meet quantitative requirements and so does the inverse. For example, when training an agent to play a game, although plenty of gameplay recordings can be obtained from the community,

action observations are missing in these recordings.

OpenAI proposed a architecture named Video Pre-Training(VPT) to solve this problem. In this architecture, a Inverse Dynamics Model(IDM) is trained on 2k hours of well labeled gameplay recordings. Then the model trained labels the missing actions of 70k hours of gameplay recordings collected from the community.[5] Therefore a data set that meets both quantitative and qualitative requirements are available for offline RL algorithm. However, since the labeling model will not be 100% accurate, the data set is likely to have errors in the action conversations which will impact the permanence of the RL model.

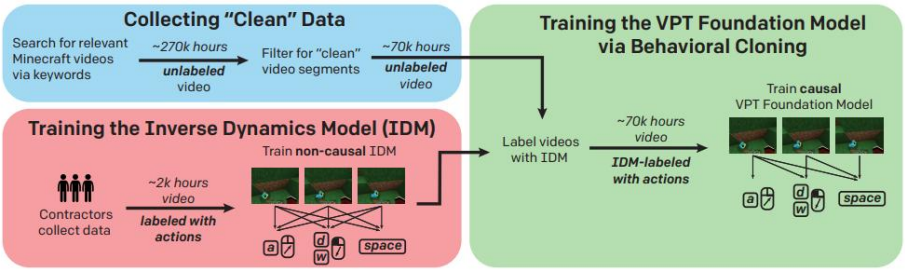


Figure 4: VPT Model Overview

Benefit from self-attention mechanism, the transformer-like architecture is able to concentrate on the important fractions. A research shows that the transformer-based model is able to keep its performance with up to 60% frame drops while the performance other RL algorithms drops significantly.[6] To further examine the performance of self-attention mechanism in non-ideal scenarios, I performed a Action Noise Test. In the test, with a 5% probability, the actions sent to the model were altered from the original actions in the data set to simulate the presence of error.

As shown in Table 2, When a relatively large error probability introduced, the Transformer Architecture still managed to maintain the same level of performance. The result indicates that the self-attention has a greater potential of resistance against noise when implementing in real world.

Table 2: Noise Test Result.

| | Hopper-medium | Walker-medium |
|---|---|---|
| With Noise | 66.1 | 73.5 |
| No Noise (official) | 67.6 | 74 |

## CONCLUSION

The transformer has multiple strength compared to other approaches such as generalization, strong performance in long credit assignment tasks, ability to utilize all data, resistant to errors and frame dropping, etc. The noise test further shows the strength of transformer based models in a non-ideal, noisy environment which has been a notable problem when

implementing RL algorithms in real world.

This project explores one aspect of possible modifications to make decision transformer architecture more friendly to implement in real world. Although there are variations to this architecture that totally abort self-attention and transformer[11], the unique strength of attention mechanism is likely lost. The experiments show that the simplification of full attention did not result in a significant loss performance unlike the tests in NLP. The results may indicate that modifications made in the flied of forecasting are applicable to decision transformer as they both need long sequence and lower memory complexity given that the simplification generally does not decrease the performance. Such modifications have the potential to be a practical choice while still keeping the advantages of self-attention mechanism mentioned in this report.

Both Reformer and Informer are expected to show its maximum strength when the sequence length gets long enough. However, due to limitations of resources and time, experiments with long sequence length are not performed. Even though the detailed performance is not clear when the sequence length gets extremely long; from the experiments performed, the replacements did not result in the obvious difference from the original transformer model even under a relative short sequence length.

## REFERENCE


[1] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems*, 2021.

[2] 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 *Neural Information Processing Systems,* 2017

[3] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[4] Yi Tay1, Mostafa Dehghani1, Samira Abnar1, Yikang Shen1, Dara Bahri1, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, Donald Metzler. Long Range Arena: a benchmark for efficient transformers. In *International Conference on Learning Representations,* 2021

[5] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, Jeff Clune. Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos. In *Neural Information Processing Systems*, 2022.

[6] Kaizhe Hu, Ray Chen Zheng, Yang Gao, Huazhe Xu. Decision Transformer under Random Frame Dropping. In *International Conference on Learning Representations*, 2023

[7] Kuang-Huei Lee, Ofir Nachum, Sherry Yang, Lisa Lee, C. Daniel Freeman, Sergio

Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, Igor Mordatch. Multi-Game Decision Transformers. In *Neural Information Processing Systems*, 2022.
[8] Nikita Kitaev, Lukasz Kaiser, Anselm Levskaya. Reformer: The Efficient Transformer In *International Conference on Learning Representations*, 2020
[9] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, Wancai ZhangInformer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI Conference on Artificial Intelligence*, 2021
[10] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. 2018
[11] Max Siebenborn, Boris Belousov, Junning Huang, Jan Peters. How Crucial is Transformer in Decision Transformer? *arXiv preprint, arXiv:2211.14655,* 2022

## Appendix A - Hyperparameters

| Hyper Parameter | DT | Reformer | Informer |
|---|---|---|---|
| Context length K | 20 | 20 | 20 |
| Number of hidden layers | 3 | 3 | 3 |
| Hidden layer size | 128 | 128 | 128 |
| Batch size | 64 | 64 | 64 |
| Dropout | 0.1 | 0.1 | 0.1 |
| Learning rate | 3e-5 | 1e-4 | 1e-4 |
| Number of attention heads | 1 | 1 | 1 |
| Number of hashes | / | 2 | / |
| Training Iterations | 100k | 20k | 20k |

## Appendix B - Code Reference

This appendix lists all the Github repositories I used in this project.
The skeleton of the code is from [2]. I only wrote the replacement of transformer with Reformer/Informer (and the modification of models needed to do this) plus the noise test. The Reformer and Informer model are from the official code release [3][4].

[1] GitHub - kzl/decision-transformer: Official codebase for Decision Transformer: Reinforcement Learning via Sequence Modeling.
[2] GitHub - nikhilbarhate99/min-decision-transformer: Minimal implementation of Decision Transformer: Reinforcement Learning via Sequence Modeling in PyTorch for mujoco control tasks in OpenAI gym
[3] GitHub - lucidrains/reformer-pytorch: Reformer, the efficient Transformer, in Pytorch
[4] GitHub - zhouhaoyi/Informer2020: The GitHub repository for the paper "Informer" accepted by AAAI 2021.
[5] GitHub - thuml/Autoformer: About Code release for "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting" (NeurIPS 2021), https://arxiv.org/abs/2106.13008