DEBRA: Decision Bidirectional Approach for Offline Reinforcement Learning

Bohdan Naida
Department of Computer Science
University of Toronto
bohdan.naida@mail.utoronto.ca

Kateryna Nekhomiazh
Department of Computer Science
University of Toronto
kateryna.nekhomiazh@mail.utoronto.ca

Mariia Rizhko
Department of Computer Science
University of Toronto
mariia.rizhko@mail.utoronto.ca

GPT and BERT are state-of-the-art natural language processing models developed in recent years. GPT is a type of language model that is trained to generate text that is similar to human-written text. BERT is a type of model that is trained to perform a variety of natural language processing tasks, such as text classification and named entity recognition. GPT and BERT have achieved impressive results on a wide range of tasks and have been widely adopted by researchers and industry practitioners. Recently Decision Transformer (GPT-based architecture) was developed for solving Reinforcement Learning tasks. In this work, we evaluate Decision Transfomer on different MuJoCo environments, investigate the training stability of the Decision Transformer, compare Decision Transformer to Conservative Q-Learning, and propose new architecture DEBRA.

1 Introduction

Reinforcement learning (RL) is an area of machine learning concentrated on learning to make decisions in order to maximize the notion of cumulative reward. RL agents are trained to optimize a specified reward signal (based on an objective) through actively interacting with the environment. The agent learns to achieve the highest reward by trial and error, which means taking action and observing the changes in the state of the environment and the reward signal. Due to the ability of Deep RL approaches to approximate high-dimensional functions [1] and [2] and the ability to allow the agents to learn the structure of the compound environments, they become more and more popular. However, what plays one of the most significant roles in the popularity and profitability of Deep RL approaches is the ability to interact with the environment and collect the trajectories. For the majority of real-world problems, this feature is not available due to the high cost, or the possibility of damage or inevitable consequences (for example, medicine). To address these concerns, a large body of work in offline reinforcement learning algorithms is proposed to use previously collected data with no further online interactions, and data collection [3, 4]. Although the RL agent is trained using the static previously-collected dataset, the goal is still achieving the highest cumulative reward in the environment.

The Transformer architecture [5] has become a standard for natural language processing tasks [6], and it is shown that they can achieve state-of-the-art performance in computer vision tasks as well [7]. Recent work has demonstrated that offline RL can also be formulated as a sequence modelling problem and can benefit from transformer-based approaches such as Decision Transformers (DTs) [8]. Despite the simplicity of the architecture of DTs, according to the authors, their performance matches or exceeds the performance of state-of-the-art offline RL approaches. The authors utilize Generative Pre-Training (GPT) [6], a decoder-only model trained using the left-to-right language objective. GPT has shown excellent results on NLP tasks. Bidirectional Encoder Representations from Transformers (BERT) [9] were introduced later. BERT used conditioning on both the left and

GitHub: https://github.com/bohdan-nd/DEBRA

the right context, which showed better performance than GPT. In this work, we test the assumption that BERT can outperform GPT in RL tasks.

While working on this approach, several inconsistencies were found in the original DT paper and codebase. We decided to take a more precise look at the implementation and evaluation. This project also addresses the fact that the results given by the authors of the DT paper can not be reproduced correctly.

The major drawback is that the performance results that the authors used for comparing and evaluating the DT were received inaccurately. For this reason, we trained the DT on different versions of gym MuJoCo environments [10], such as HalfCheetah, Hopper, and Walker2d. Compared their performance with the current state-of-the-art approach – Conservative Q-learning [11]. We also investigated the stability of the DT, as it was claimed that their performance during the training was stable, allowing us to compare and evaluate it correctly.

2 Related Work

Generative Pre-trained Transformer (GPT) is a type of language model that uses a deep learning technique called transformer architecture to generate text. It is pre-trained on a large corpus of text data and can then be fine-tuned on specific tasks such as language translation or text summarization. The GPT model uses self-attention mechanisms to process the input text and generate output that is coherent and mimics human-like language patterns. It is a powerful tool for natural language processing tasks and has achieved state-of-the-art results on a variety of benchmarks.

Bidirectional Encoder Representations from Transformers (BERT) is trained on a large corpus of text data and uses a bidirectional approach, which means that it takes into account the context of the words in a sentence from both the left and the right sides. This allows BERT to understand the meaning and context of words in a sentence more accurately than many other language models. It can then be fine-tuned on specific tasks such as sentiment analysis or named entity recognition. BERT has achieved state-of-the-art results on many natural language processing tasks and is widely used in the industry.

Decision Transformer (DT) is a framework that allows bringing up RL tasks as a sequence modelling problem. This approach allows using language modelling, such as GPT, in the RL domain. DT generates actions based on future desired returns and represents trajectories as $R_1, s_1, a_1, R_2, s_2, a_2..., R_n, s_n, a_n$, where R_t is a reward, S_t is a state, and a_t is an action on timestep t. Given the last K timesteps, the DT model predicts the future action token via autoregressive modelling.

Conservative Q-learning (CQL) is a variant of the popular Q-learning algorithm, a reinforcement learning technique used to find the optimal action-selection policy for an agent interacting with its environment. The main difference between traditional Q-learning and conservative Q-learning is that the latter uses a more conservative approach to updating the Q-values or the estimates of the expected return of each action.

3 Method

In this work, we propose the **DE**cision **B**idi**R**ectional **A**pproach (DEBRA), a new decision transformer architecture based on the bidirectional transformer encoder [9]. We use the BERT architecture to predict the next token in a sequence of returns-to-go (sum of future rewards), states, and actions.

Trajectory: In our method, we treat trajectory data as a sequence for modelling by a Transformer architecture. Since the model's goal is to generate an action based on future potential rewards, we use the returns-to-go as in the original DT [8]: $\hat{R} = \sum_{t=t}^{T} r_t$.

A trajectory τ consists of T returns-to-go, states, and actions: $T = (\hat{R_1}, s_1, a_1, \hat{R_2}, ..., \hat{R_t}, s_t, a_t)$

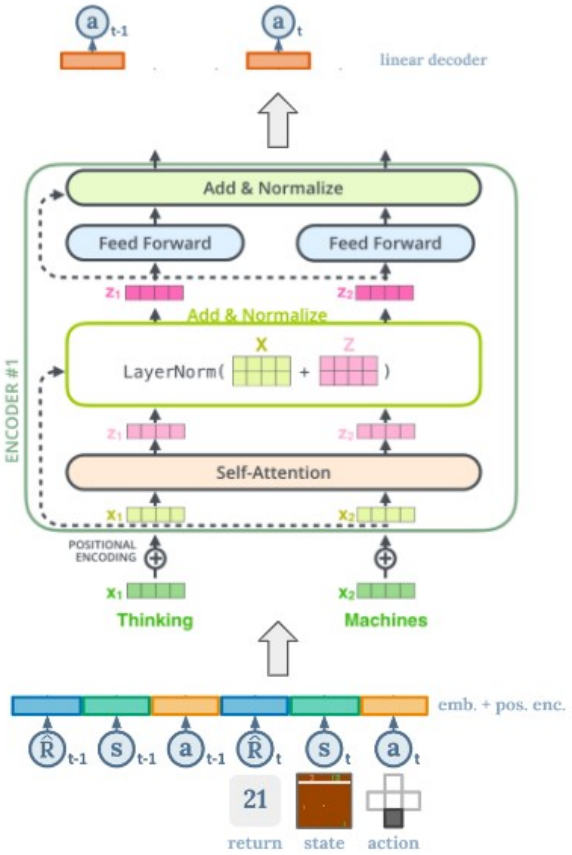

Figure 1: DEBRA Architecture

Architecture: The DEBRA Architecture is shown in Figure 1. First, DEBRA transforms the raw input of states, actions, and rewards into tokens using linear embedding layers. Then, it changes the data format to the trajectory format described above. The encoder model in our architecture is a Transformer decoder mirroring the BERT architecture. We trained DEBRA on a smaller architecture than typically used in large-scale language modelling, consisting of one layer and three self-attention heads. DEBRA has a bidirectional nature, so during the training, the masking mechanism relies on tokens from both sides. This feature allows DEBRA to predict an action based not only on a previous trajectory but also on the following one during the training stage. Moreover, because of BERT's Next Sentence Prediction stage, DEBRA is trained to understand a relationship between two sub-trajec0tories. DEBRA's masking mechanism can improve the trajectories from the dataset by selecting a more optimal state-action pair instead of the masking one.

4 Evaluation

We used three environments for evaluation: Walker2d, HalfCheetah, and Hopper. They are all environments in the OpenAI Gym, a collection of environments for developing and testing reinforcement learning algorithms. These environments are designed for use with robotic control tasks, and each one simulates a different type of robotic system.

- The **Walker2d** environment simulates a two-legged robot. The goal of the Walker2d environment is for the robot to coordinate both legs to move forward.

Dataset	Environment	DT(Ours)	DT (Original paper)
Medium-Expert	Half Cheetah	89.53	86.80
	Hopper	102.87	107.60
	Walker	107.76	108.10
Medium	Half Cheetah	42.28	42.60
	Hopper	60.42	67.60
	Walker	33.20	**74.00**
Medium-Replay	Half Cheetah	29.74	36.60
	Hopper	47.36	**82.70**
	Walker	71.02	66.60

Table 1: Results for DT reproduced by us and stated in the original paper

- The **HalfCheetah** environment simulates a robotic cheetah that can run and manoeuvre in different directions. The goal of the HalfCheetah environment is for the robotic cheetah to move as quickly as possible in a straight line.
- The **Hopper** environment simulates a one-legged hopping robot. The goal of the Hopper environment is for the robot to hop forward as quickly as possible without falling over.

These environments can be used to train reinforcement learning algorithms to control robotic systems, and they provide a challenging testbed for evaluating the performance of different algorithms. We evaluated results on "random", "medium", "medium-replay", "medium-expert", and "expert" datasets for each of these environments. We used both version 0 (further Walker2d-v0, HalfCheetah-v0, and Hopper-v0) and version 2 (further Walker2d-v2, HalfCheetah-v2, and Hopper-v2) for comparison.

5 Results

5.1 DT results on Walker2d, HalfCheetah, and Hopper

We reproduced the code from the original DT paper and trained the model on "random", "medium", "medium-replay", "medium-expert", and "expert" datasets of Walker2d-v2, HalfCheetah-v2, and Hopper-v2 environments. We used the same hyperparameters that were used in the original paper. The results are shown in Table 1.

Most results are similar to the ones stated in the original paper, but DT trained on Walker-v2 Medium and Hopper-v2 Medium-Replay have a significant difference in scores. We investigate why this happened in section 5.2.

5.2 Stability

Given the inconsistent results of our implementation of DT and the results stated in the original paper, we discovered that DT training process isn't stable. The average return goes up and down during the training instead of consistently increasing. The average return for each epoch for some experiments can be found in Figure 2. The plots are made using the Weights and Biases platform [12].

This is a problem because it can make it difficult to properly evaluate the model's performance and determine whether it is making accurate decisions. In Table 2, we show the results from the best epoch (based on return) instead of the last epoch. When selecting the best epoch instead of the last epoch for the Walker-v2 Medium and Hopper-v2 Medium-Replay experiments, we received similar results stated in the original paper.

This instability may be due to several factors, including the complexity of the decision-making task and the quality of the training data. To address this issue, we may need to employ various techniques,

(a) Hopper-v0 Medium-replay (b) Hopper-v2 Medium (c) Walker-v2 Random

Figure 2: Unstable training process

Dataset	Environment	DT(Ours)	DT (Original paper)
Medium-Expert	Half Cheetah	89.53	86.80
	Hopper	102.88	107.60
	Walker	107.77	108.10
Medium	Half Cheetah	42.55	42.60
	Hopper	65.70	67.60
	Walker	**70.02**	74.00
Medium-Replay	Half Cheetah	36.01	36.60
	Hopper	**74.24**	82.70
	Walker	71.02	66.60

Table 2: Results for DT reproduced by us (the best epoch) and stated in the original paper

such as regularization or optimization methods, to improve the stability of the Decision Transformer during training. This will be in the scope of future work.

5.3 Comparison of DT to CQL

It sounds like the paper's authors may have made a mistake by comparing the results of their experiment using different versions of an environment or dataset. This can be a problem because the results of an experiment can be influenced by the specific characteristics of the environment or dataset used. If the environment or dataset is changed, the results of the experiment or analysis may not be comparable to those obtained from the original environment or dataset.

The authors of CQL showed results on Walker2d-v0, HalfCheetah-v0, and Hopper-v0, while the authors of DT showed results on Walker2d-v2, HalfCheetah-v2, and Hopper-v2. For a fair comparison of these algorithms, we trained DT on Walker2d-v0, HalfCheetah-v0, and Hopper-v0. As previously discussed problems with instability of training, we report the results both for the last and the best epoch for DT. The results are shown in Table 3.

The results show that CQL outperforms DT almost in all datasets for each environment. DT exceeds CQL in Hopper Medium-Expert, Hopper Medium, and Hopper Expert only if we select the score from the best epoch. The only case when DT outperforms CQL is in the Half Cheetah Medium-Expert environment; DT reached a score of 83.9, while CQL - only 62.4.

6 Limitations

Significant restraint of the DT is an inability to perform reliably in stochastic environments [13]. Because the stochasticity of the world influences almost every real-world task, we cannot rely on DT. DT and other methods that condition on outcomes such as return [14] [15] can take suboptimal actions in stochastic environments disregarding scale or the amount of data they are trained on.

Dataset	Environment	DT (Last epoch)	DT (Best epoch)	CQL
Random	Half Cheetah	2.2	2.3	**35.4**
	Hopper	10.1	10.2	**10.8**
	Walker	0.1	5.9	**7.0**
Medium-Expert	Half Cheetah	83.9	**83.9**	62.4
	Hopper	102.9	**111.8**	111.0
	Walker	15.6	81.0	**98.7**
Medium	Half Cheetah	36.4	37.2	**44.4**
	Hopper	32.0	**82.2**	58.0
	Walker	3.3	16.3	**79.2**
Medium-Replay	Half Cheetah	33.2	33.5	**46.2**
	Hopper	8.5	30.6	**48.6**
	Walker	11.4	20.8	**26.7**
Expert	Half Cheetah	96.1	98.0	**104.8**
	Hopper	108.9	**112.2**	109.9
	Walker	13.5	78.6	**153.9**

Table 3: Comparison of DT and CQL for environments with version 0

When the model conditions on trajectories that result in receiving a positive reward, it doesn't "learn" any cases where the same trajectories lead to a negative reward.

DEBRA, as Decision Transformers in general, has a lack of explainability. The model is a black box that transforms raw trajectory input into token embeddings and maps those into other tokens—the nature of decision-making.

7 Conclusion

We hypothesized that the DTs based on BERT could outperform the original DTs due to their bidirectional nature. In this project, we proposed a **DE**cision **B**idi**R**ectional **A**pproach (DEBRA). DEBRA has not shown a higher performance than the original model. While researching the potential reasons, we found that the original paper's authors made a few mistakes, such as the models' training and evaluation were done in different environments. After fixing inconsistencies, we found that the original DT performs worse than the previous state-of-the-art approach - CQL. The DT training process is volatile: the mean of returns can decrease more than twice. Additionally, the DT architecture is a black box, so it lacks explainability and cannot perform reliably in stochastic environments. To conclude, transformer architecture is not a panacea, and in the RL domain, approaches such as CQL are more valuable. We plan to fix DT instability and poor performance of DEBRA in future work.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[3] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[4] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[9] 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.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

[12] L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

[13] K. Paster, S. McIlraith, and J. Ba. You can't count on luck: Why decision transformers fail in stochastic environments. *arXiv preprint arXiv:2205.15967*, 2022.

[14] S. Emmons, B. Eysenbach, I. Kostrikov, and S. Levine. Rvs: What is essential for offline RL via supervised learning? *CoRR*, abs/2112.10751, 2021. URL https://arxiv.org/abs/2112.10751.

[15] J. Schmidhuber. Reinforcement learning upside down: Don't predict rewards - just map them to actions. *CoRR*, abs/1912.02875, 2019. URL http://arxiv.org/abs/1912.02875.

A Appendix

Equal contribution. Bohdan worked on designing and training DEBRA and DTs based on various transformer architectures, analyzing the performance. Kateryna worked on running the experiments, analyzing the DT performance, comparing the results for different configurations, and creating visualizations. Mariia worked on analyzing DT architecture, writing code for training, and comparing results for different environments (version 0 vs version 2) and methods (CQL). Everyone participated in preparing a presentation and writing reports.