SACFormer: Enhancing Soft Actor-Critic with Sequence Modeling

Yuan Qing [1] **Tianyi Xu** [1] **Yida Wang** [1]

Abstract

In the realm of Reinforcement Learning (RL), Decision Transformer (DT) approaches have shown promise across various control tasks. However, a significant challenge lies in their predominant reliance on offline training, making direct online application of DT methods less feasible. Notably, online methods generally employ an actor-critic framework. In response to the difficulties encountered in online DT training, we introduce SACFomer, an innovative online RL framework that integrates sequence modeling. SACFomer uniquely incorporates a DT into the actor network within an actor-critic architecture. This approach has yielded competitive results in various OpenAI Gym environments, outperforming several baseline methods. An ablation study further elucidates the benefits of sequence modeling and investigates the impact of sequence length. The code of our project is available at *https://github.com/XTTTZ/SACFormer*.

1. Introduction

Recently , transformer architecture (Vaswani et al., 2017) has brought about a revolutionary change in deep learning (DL), particularly in the realms of natural language processing (NLP) (Devlin et al., 2018; Floridi & Chiriatti, 2020) and computer vision (CV) (Dosovitskiy et al., 2020; Han et al., 2022). This paradigm shift is attributed to the transformer's unparalleled proficiency in modeling complex, high-dimensional distributions of semantic concepts on a large scale. Its unique structure, characterized by self-attention mechanisms, allows for more efficient and effective handling of sequential data compared to traditional models. Inspired by the applications of transformer architecture in NLP and CV, the Decision Transformer (DT) (Chen et al., 2021) has been designed, tailoring this technology to the realm of RL through reconceptualizing RL as a sequence modeling problem. DT enables the exploitation of the transformer architecture's inherent proficiency in processing sequential data, adapting it for decision-making tasks.

Compared to traditional temporal difference (TD) learning (Sutton & Barto, 2018) based approaches, DT based methods have demonstrated impressive performance across a variety of offline reinforcement learning (RL) benchmarks (Li et al., 2023; Lee et al., 2022; Meng et al., 2021; Xu et al., 2022). However, a theoretical understanding of how DT operates in RL remains limited. This is particularly pertinent in the context of a Markov Decision Process (MDP), where the relevance of past experiences to the probability density of the current policy is not inherently necessary. In addition, the majority of DT based methods are confined to offline RL. Offline RL trains RL algorithms on a fixed dataset of previously collected experiences, without further interacting with the environment (Fu et al., 2020; Levine et al., 2020; Prudencio et al., 2023). This focus on offline RL presents several drawbacks, including potential overfitting to the training data, limited exploration of the environment, and challenges in generalizing to unseen scenarios (Kumar et al., 2019; Agarwal et al., 2020).

Given the compelling performance exhibited by DT, our goal is to address and mitigate the inherent limitations of their offline nature by exploring new methodologies. In contrast to previous approaches that involve initial offline pretraining followed by online fine-tuning (Xie et al., 2022; Meng et al., 2021), we aim to investigate the feasibility of developing a purely online RL architecture based on DT. This exploration intends to harness the strengths of DT while effectively adapting it for dynamic, real-time learning environments.

In this paper, we elucidate that the goal can be attained by adopting an actor-critic-like structure, seamlessly integrated with DT. To model an offline RL schema using DT, we examine the structures of traditional actor-critic methods, which have established well-defined theoretical bounds in policy optimization (Lillicrap et al., 2015a; Mnih et al., 2016; Kumar et al., 2020; Haarnoja et al., 2018b; Fujimoto et al., 2018). Notably, a significant portion of recent actor-critic methods are based on the Soft Actor-Critic (SAC) framework (Haarnoja et al., 2018b). Building upon this, we propose a novel online RL approach **SACFormer** that

[1]Department of Electrical Engineering, Columbia University, New York, USA. Correspondence to: Yuan Qing <yq2379@columbia.edu>, Tianyi Xu <tx2237@columbia.edu>, Yida Wang <yw4082@columbia.edu>.

adapts the SAC architecture, integrating DT within the actor network to enhance its performance and applicability in online learning scenarios. To the best of our knowledge, there is no existing research on purely online DT that have been integrated with actor-critic architecture. Our contributions can be summarized as follows:

- As far as we know, we are the first to define and tackle pure online RL tasks using transformer architecture.
- We design a novel online RL framework that innovatively amalgamates sequence modeling capabilities with SAC.
- We present empirical results that show that SACFormer achieves competitive results in Open AI Gym (Brockman et al., 2016) environments compared with other baseline methods.

2. Related Work

2.1. Online RL

Online RL is an important branch in the field of machine learning that mimics the process by which humans or animals learn to make decisions by interacting with their environment. In this process, an agent continuously adjusts its behavioral strategy by performing actions and receiving feedback (reward or punishment) from the environment to maximize long-term rewards (Zhao et al., 2023).

This approach is opposed to offline RL, which collects data while the intelligent body interacts with the environment and then trains on this data, rather than learning directly from real-time environmental interactions. The advantage of online reinforcement learning is its immediacy and continuity (Ball et al., 2023). Instead of waiting for a large amount of data to accumulate before batch learning, agencies learn from each interaction with their environment in real time, which allows them to adapt to changes in the environment more quickly. In addition, this type of learning reduces the reliance on large amounts of historical data, and agencies can make effective decisions in unforeseen new environments.

Online RL is efficient for its sample efficiency. Online RL can efficiently utilize off-policy data for fine-tuning, allowing agents to learn from a continuous stream of experiences. This enhances sample efficiency, which is vital in environments where data collection is expensive or difficult (Liu et al., 2023). Other than that, it balances exploitation and exploration pretty well. Online RL provides methods for more intelligent exploration strategies beyond traditional approaches like ϵ-greedy, improving the ability to explore action spaces in complex environments and preventing the agent from getting stuck in local optima (Eysenbach et al., 2020). Plus, it keeps learning and refining itself from visual observations in the environment. (Laskin et al., 2020) proposes an online RL method that incorporates techniques like data augmentation to improve both the data efficiency of learning and generalization to new environments, crucial for tasks that require perceiving high-dimensional inputs. Compared with previous works, we focus on forming an online RL framework in the context of sequence modeling in this project.

2.2. Transformer

The original version of Transformer was designed to improve machine translation(Vaswani et al., 2017). The Transformer relies on an attention mechanism to draw global dependencies between input and output, breaking away from the traditional recurrent or convolutional networks. This design enables the model to capture long-range dependencies in input data. The Multi-head self-attention feature allows the model to process information from different representation subspaces at different positions, enhancing its ability to understand various aspects of the input.

To expand Transformer to more realms in NLP, the Bidirectional Encoder Representations from Transformers (BERT)(Devlin et al., 2018) was introduced. BERT's key innovation lies in its ability to consider the full context of a word by looking at both the left and the right side of the word in a text. This bidirectional context understanding contrasts with previous models that typically processed text in one direction(Peters et al., 2018; Radford & Narasimhan, 2018). BERT is built upon the Transformer architecture, specifically utilizing its encoder mechanism. The Transformer model, known for its attention mechanism, allows BERT to weigh the importance of different words in the context effectively.

The Vision Transformer (ViT) was introduced to split the image into small patches and feeding into Transformer model (Dosovitskiy et al., 2020). Facebook AI came up with Data-efficient Image Transformers (DeiT) (Touvron et al., 2021). DeiT optimizes for data efficiency, training Transformers effectively with less data and a novel distillation strategy. Swin Transformer (Liu et al., 2021) is ahierarchical Transformer architecture with variable-sized windows for self-attention, making it more adaptable to images of varying sizes and resolutions. Convolutional vision Transformers (CvT) (Wu et al., 2021) combines the benefits of CNNs and Transformers by introducing convolutional operations into the Transformer architecture, enhancing local feature capture while retaining long-range dependencies.

The Decision Transformer, an innovative approach that applies transformer-based architectures, typically used in natural language processing, to reinforcement learning tasks(Chen et al., 2021). This model represents a significant departure from traditional reinforcement learning methods by treating the reinforcement learning problem as a sequence modeling task. The Decision Transformer takes in a sequence of states, actions, and rewards, and predicts the next action, conditioned on a desired return or reward to go. Trajectory Transformer (Janner et al., 2021) uses Transformers to model entire trajectories in the reinforcement learning context, focusing on accurate behavioral cloning and offline policy optimization. GPT for Finance (GPT-f) (Wallbridge, 2020) adapts the Decision Transformer approach for financial decision-making tasks, using Transformers to predict financial actions and strategies based on historical financial data. Compared with other DT based methods, we focus on applying DT to online RL under an actor-critic framework.

2.3. Actor-Critic

Actor-Critic is a reinforcement learning algorithm that combines two main reinforcement learning methods: Policy Gradient and Value Function Approximation. In this framework, the ”actor” is responsible for learning what actions to take, while the ”criterion” evaluates how good or bad these actions are, i.e., gives an estimate of their value(Haarnoja et al., 2018a). This structure allows the algorithm to efficiently balance the trade-off between exploration (trying new actions) and exploitation (utilizing the best known actions).

In the Actor-Critic approach, policy optimization is performed in a continuous and progressive manner, which ensures the stability of the learning process. This approach reduces performance fluctuations due to policy changes, which is especially important in long-term reinforcement learning tasks. A stable learning process allows the algorithm to gradually adapt to changes in the environment and continuously improve its strategy(Achiam, 2018).

The Actor-Critic algorithm places special emphasis on the efficient use of samples in the learning process. Through the experience replay mechanism, the algorithm can learn from past experiences rather than relying only on recent data. This approach improves the efficient use of data and speeds up the learning process, especially in complex environments(Andrychowicz et al., 2018).

Another key feature of the Actor-Critic method is that it is applicable to various types of action spaces, both discrete and continuous. This allows it to be flexibly applied to different types of reinforcement learning problems, from simple games to complex robot control. In continuous action spaces, the Actor-Critic method is particularly effective because it is able to output a continuous action value directly, rather than selecting from a limited set of actions(Haarnoja et al., 2018b).

The combination of the Actor-Critic approach with deep learning techniques makes it possible to process high dimensional input data such as images and complex sensor data. Deep learning networks (e.g., convolutional neural networks or recurrent neural networks) are often used as part of the Actor and Critic modules to process these complex inputs. This integration allows Actor-Critic algorithms to learn effective strategies in visually complex environments. Most actor-critic methods are predominantly dependent on deep neural networks, with limited exploration into the use of transformer architectures. In this project, we augment the actor network by integrating a transformer model, thereby enriching the Soft Actor-Critic (SAC) framework with the capability to leverage sequential information.

3. Proposed Method

The following sections give detailed explanations of our proposed method *SACFormer* step by step. We first introduce the adapted online RL framework SAC. Then, we give an overall illustration on the DT structure and specifically our modifications to the DT architecture. Finally, we elaborate on our method in detail.

3.1. Decision Transformer

Following DT (Chen et al., 2021), we introduce sequence modeling to the actor network through considering a sequence of return-to-go and states. Here we denote G_t (return-to-go) as the accumulated discounted reward at state S_t. In SAC (Haarnoja et al., 2018b), the policy network (actor) produces the mean and standard deviation of the Gaussian distribution when using the Gaussian policy. Following this paradigm, we add two heads to the transformer output to regress the two values instead of the original DT implementation which directly predicts the next action A_{t+1} based on all the previous return-to-gos, states and rewards

$$\{(G_1, S_1, A_1), (G_2, S_2, A_2), \cdots (G_t, S_t, A_t)\}.$$

In contrast, we use a sequence of

$$\{(G_1, S_1), (G_2, S_2), \cdots (G_t, S_t)\}$$

to predict the mean and standard deviation of the Gaussian distribution at each state. To be specific, during training, a sequence containing K tuples of return-to-go and state will be sampled from the replay buffer (Mnih et al., 2013). This sampled sequence will be the input of the GPT-2 model, i.e., the decoder block. In this way, we first embed G_t and S_t to the same dimension and then stack them along the sequence dimension to form a sequence as follows

$$(G_1, S_1, G_2, S_2, G_3, S_3, \cdots, G_K, S_K),$$

which then has a length of $2K$. In addition to state and return-to-go embeddings, we manually added a positional embedding to each step in the sequence in time dimension. Similarly for evaluation, we start from timestep 0 with a state and preset return-to-go G_1 (this is a manually set constant and is task-specific). For every future steps, we update the return-to-go as follows

$$G_t = G_{t-1} - R_{t-1}, \tag{1}$$

where R_{t-1} is the instant reward received at step $t-1$. Following this pipeline, we can then obtain the expected output in an autoregressive way. Note that we only use the second dimension of the decoupled output hidden states as the input to MLP heads as indicated by the solid black line in Figure 1. In the design of the critic network, we intentionally omit the incorporation of sequence information. This decision is grounded in the objective of enhancing the stability of the network's output. By excluding temporal or sequential dependencies from the critic's architecture, we aim to mitigate potential volatility in its predictions, thereby fostering a more consistent and reliable evaluation of state-action pairs.

3.2. Soft Actor-Critic

MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, p, r)$. The space state $\mathcal{S}$ and action space $\mathcal{A}$ are continuous. The state transition probability $p:\ \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,\, \infty)$ maps the current state $S_T \in \mathcal{S}$ and action $A_T \in \mathcal{A}$ to the probability density of the next state $S_Tp \in \mathcal{S}$. On each transition, The environment emits a reward $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$. We will also use $\rho_\pi(S_T)$ and $\rho_\pi(S_T, A_T)$ to denote the state and state-action marginals of the trajectory distribution induced by a policy $\pi(A_T|S_T)$.

The SAC aims to learn a policy $\pi(A_T|S_T)$ that maximizes the expected sum of rewards $\sum_t \mathbb{E}(S_T, A_T) \sim \rho_\pi r(S_T, A_T)$, extended by the maximum entropy objective (see (Ziebart, 2010) which also maximizes entropy at each visited state. This is expressed as:

$$\begin{aligned}\pi^* = &\arg\max_\pi \sum_t \mathbb{E}_{(S_T,A_T)\sim\rho_\pi}\big[r(S_T, A_T) \\ &+ \alpha\mathcal{H}(\pi(\,\cdot\,|S_T))\big],\end{aligned} \tag{2}$$

where α is the temperature parameter that determines the relative importance of the entropy term versus the reward, and thus controls the stochasticity of the optimal policy.

To tackle large continuous domains, we employ function approximators for both the soft Q-function $Q_\theta(S_T, A_T)$ and the policy $\pi_\phi(A_T|S_T)$, alternating between optimizing both networks with stochastic gradient descent. The parameters of these networks are θ and ϕ. We will next derive update rules for these parameter vectors following (Haarnoja et al., 2018b).

Here we first denote a sequence of states and discounted returns (i.e., return-to-gos)

$$S_T = \{s_1, s_2, s_3, \cdots, s_T\}, \tag{3}$$

where T is the current timestep in trajectory of an episode. Similarly, we denote A_T and G_T as

$$A_T = \{a_1, a_2, a_3, \cdots, A_T\}, \tag{4}$$

$$G_T = \{g_1, g_2, g_3, \cdots, g_T\}. \tag{5}$$

Then, the soft Bellman residual under sequential contexts can be written as

$$\begin{aligned}J_Q(\theta) = &\mathbb{E}_{(S_T,A_T)\sim\mathcal{D}}\Bigg[\frac{1}{2}\Big(Q_\theta(S_T, A_T) - \big(r(S_T, A_T) \\ &+ \gamma\,\mathbb{E}_{S_{T+1}\sim p}\left[V_{\bar\theta}(S_{T+1})\right]\big)\Big)^2\Bigg],\end{aligned} \tag{6}$$

where the value function is implicitly parameterized through the soft Q-function parameters via

$$V(S_T) = \mathbb{E}_{A_T\sim\pi}\left[Q(S_T, A_T) - \alpha\log\pi(A_T|S_T)\right], \tag{7}$$

and it can be optimized with stochastic gradients

$$\begin{aligned}\hat\nabla_\theta J_Q(\theta) = \nabla_\theta Q_\theta(S_T, A_T)\Big[Q_\theta(S_T, A_T) - \Big(r(S_T, A_T) \\ +\gamma\Big(Q_{\bar\theta}(S_{T+1}, A_{T+1}) - \alpha\log\left(\pi_\phi(A_{T+1}|S_{T+1})\right)\Big)\Big].\end{aligned} \tag{8}$$

The update makes use of a target soft Q-function with parameters $\bar\theta$ that are obtained as an exponentially moving average of the soft Q-function weights, which has been shown to stabilize training (Mnih et al., 2015). Finally, the policy parameters can be learned by directly minimizing the expected KL-divergence in

$$\pi_{\text{new}} = \arg\min_{\pi'\in\Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|S_T)\,\middle\|\,\frac{\exp\left(\frac{1}{\alpha}Q^{\pi_{\text{old}}}(S_T,\,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(S_T)}\right). \tag{9}$$

(multiplied by α and ignoring the constant log-partition function and by α):

$$\begin{aligned}J_\pi(\phi) = &\mathbb{E}_{S_T\sim\mathcal{D}}\Big(\mathbb{E}_{A_T\sim\pi_\phi}\big(\alpha\log\left(\pi_\phi(A_T|S_T)\right) \\ &- Q_\theta(S_T, A_T)\big)\Big)\end{aligned} \tag{10}$$

Numerous strategies exist to minimize J_π. Commonly in policy gradient methods, the likelihood ratio gradient estimator (Williams, 1992) is employed. This method circumvents the need for backpropagation through both the policy and the target density networks. However, in our scenario, where the target density is represented by the differentiable Q-function within a neural network, employing the reparameterization trick is more advantageous. This approach offers the benefit of a lower variance in the estimator compared to traditional methods. To that end, we reparameterize the policy using a neural network transformation

$$A_T = f_\phi(\epsilon_t; S_T, G_T), \tag{11}$$

where ϵ_t is an input noise vector, sampled from some fixed distribution, such as a spherical Gaussian. We can now rewrite the objective in Equation 10 as

$$\begin{aligned} J_\pi(\phi) =& \mathbb{E}_{S_T \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \Big[\alpha \log \pi_\phi(f_\phi(\epsilon_t; S_T)|S_T) \\ & - Q_\theta(S_T, f_\phi(\epsilon_t; S_T)) \Big], \end{aligned} \tag{12}$$

where π_ϕ is defined implicitly in terms of f_ϕ. We can approximate the gradient of Equation 12 with

$$\begin{aligned} \hat{\nabla}_\phi J_\pi(\phi) =& \nabla_\phi \alpha \log\left(\pi_\phi(A_T|S_T)\right) \\ & + \Big(\nabla_{A_T} \alpha \log\left(\pi_\phi(A_T|S_T)\right) \\ & - \nabla_{A_T} Q(S_T, A_T) \Big) \nabla_\phi f_\phi(\epsilon_t; S_T), \end{aligned} \tag{13}$$

where A_T is evaluated at $f_\phi(\epsilon_t; S_T)$. The DDPG style policy gradients (Lillicrap et al., 2015b) is extended to any tractable stochastic policy through this unbiased gradient estimator. A detailed description and implementation details of the SAC can be found in (Haarnoja et al., 2018b).

3.3. SACFormer

In this section, we explain our method in detail concerning the algorithm structure. The overall structure is depicted in Figure 1. The SACFormer has an overall structure similar to the actor-critic method with a critic network implemented using a deep neural network and actor network modeled in the form of decision transformer. A replay buffer is constructed for online RL. However, compared with the ordinary replay buffer that stores the experience tuple for each environment step, our replay buffer stores the trajectory of each environment episode to maintain the sequential information. Compared with vanilla SAC, we output the actions autoregressively based on our previously seen states and return-to-gos, the detailed algorithm is illustrated in Algorithm 1 and Algorithm 2.

4. Experiments

The objective of our experiments is to explore the efficacy of integrating DT into an online SAC framework. We compare SACFormer against existing online actor-critic methods on a set of challenging continuous control tasks from the OpenAI

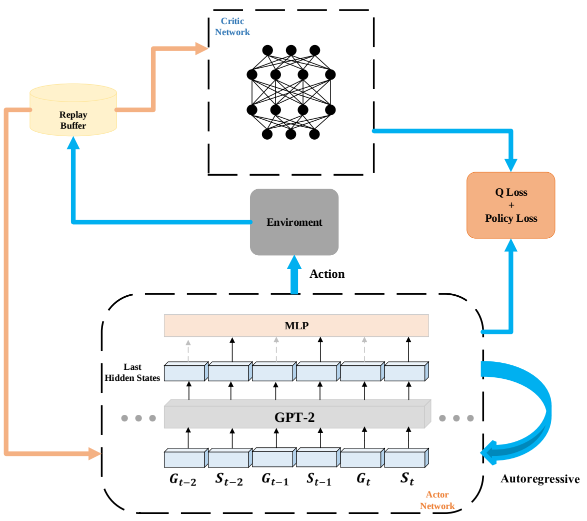


Figure 1. Structure of proposed method SACFormer. An actor network is formed based on DT (Chen et al., 2021) by taking the discounted returns and states as input. A critic network is constructed based on a simple neural network. The policy loss and Q loss are jointly optimized.

gym benchmark suite (Brockman et al., 2016). In addition, we scrutinize the impact of varying sequence lengths on the model's performance. In the following sections, we provide detailed insights into the implementation and experimentation of our method.

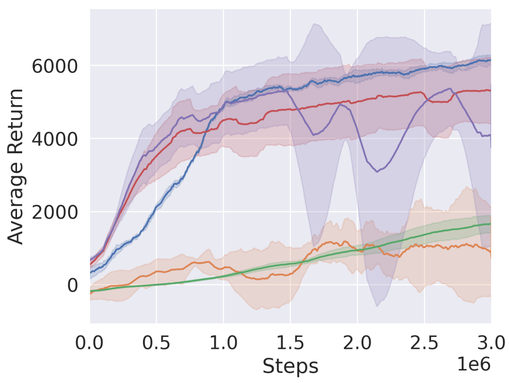
4.1. Implementation Details

Environments. We use five environments from OpenAI gym benchmark suite (Brockman et al., 2016) which are Walker2d-v2, Humanoid-v2, HalfCheetah-v2, Ant-v2 and Hopper-v2. We refer readers to (Fu et al., 2020) for detailed information about the environments.

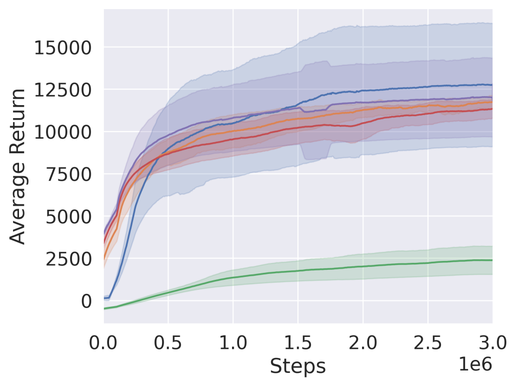
Model Architecture. We use a GPT-2 (Radford et al., 2019) model as the backbone of actor network with three layers and a hidden state dimension of 128. For the critic network and the target critic network, we follow the setting of Q network directly from (Haarnoja et al., 2018b).

Hyperparameter and Training Setting The dropout rate of the GPT-2 model is set to 0.1 and the learning rate is set to 0.0003. The discount factor γ for calculating return-to-goal is set to 0.99. We use soft parameter update (Lillicrap et al., 2015a) to update the target network with a target smoothing coefficient τ of 0.005. The training batch size is set to 128 and we use the Adam (Kingma & Ba, 2014) optimizer for both actor and critic. During training, the sequence length K is set to 6 for all the environments.

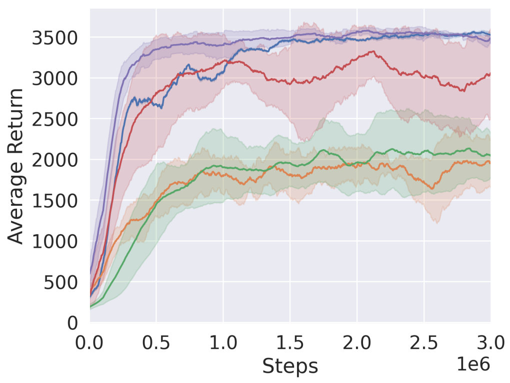
For more details on the implementation of our method,



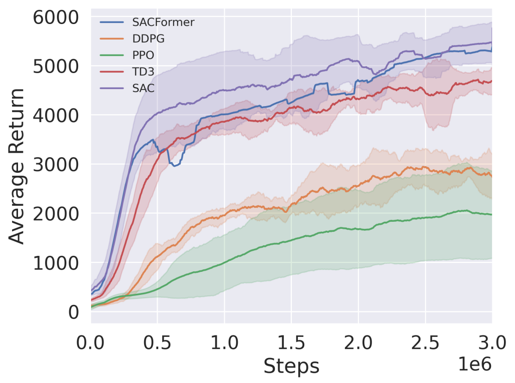
(a) Ant-v2



(b) HalfCheetah-v2



(c) Hopper-v2



(d) Walker2d-v2

Figure 2. Average episode return on four Open AI Gym tasks of SACFormer and four other baseline methods. SACFormer (blue) achieved competitive results in all the environments.

please refer to our GitHub repository.

4.2. Baselines

In this section, we present several baseline methods in online RL against which we evaluate the performance of our proposed methods.

Deep Deterministic Policy Gradient (DDPG): DDPG (Lillicrap et al., 2015a) is a model-free algorithm that merges DPG and DQN concepts, excelling in continuous action spaces. It uses a policy network for action selection and a Q-network for evaluation, incorporating off-policy learning and a replay buffer for efficiency.

Proximal Policy Optimization (PPO): PPO (Schulman et al., 2017) is a policy gradient method known for its simplicity and effectiveness, using a clipped objective to ensure moderate policy updates. It's well-suited for various environments, particularly those with complex observation spaces.

Twin Delayed DDPG (TD3): TD3 (Fujimoto et al., 2018) improves upon DDPG by introducing twin Q-networks and a delayed policy update, reducing overestimation bias. This makes it more stable and reliable for continuous control tasks.

Soft Actor-Critic (SAC): SAC (Haarnoja et al., 2018b) optimizes a stochastic policy in an entropy-enhanced reward framework, balancing expected return and exploration. Its efficiency and robustness make it a preferred choice for tasks needing strong exploration tactics.

4.3. Results

For experiments on all the environments, we report the average episode return on 4 testing episodes every 2000 training steps and all the results are averaged over three different random seeds. We follow (Li, 2021) to report the benchmark results on TD3, DDPG and PPO. The results are visualized in Figure 2. From the results, we can see that our SACFormer outperforms all other methods in two environments *Ant-v2* and *HalfCheetah-v2*. In *Hopper-v2*, it converges to almost the same average episode return

Algorithm 1 SACFormer (Training)

```
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\pi(s|\theta^\pi)$ with weights $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ with weights $\theta^{Q'} \leftarrow \theta^Q$.
Initialize trajectory replay buffer $\mathcal{R}$.
repeat
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  Initialize a trajectory array $\mathcal{T}$
  repeat
    Select actions $a_t = \pi(S_T, G_T|\theta^\pi) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$
    Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{T}$
    for 1 to #Gradient update steps do
      Sample a random minibatch of $N \times K$ steps $(s_i, a_i, r_i, s_{i+1})$ from $N$ trajectories in $\mathcal{R}$
      Predict next step actions $A_{T+1}$ and next $\log \pi$ using actor network.
      Calculate next step q-values $q_{T+1}$ using critic network.
      Calculate current step q-values $q_T$.
      Calculate critic q-loss $MSE(q_T, q_T + 1)$.
      Predict current step actions $A_T$ using $S_T$ and $G_T$.
      Predict q-values of the predicted actions using actor network.
      Calculate policy loss using Equation 13.
      Backpropagation; update parameters.
    end for
  until done
  Push $\mathcal{T}$ to $\mathcal{R}$.
until Max total steps
```

Algorithm 2 SACFormer (Inference)

```
Initialize actor network $\pi(s|\theta^\pi)$ with trained weights $\theta^\pi$.
Receive initial observation state $s_1$.
Set initial return-to-go $G_1$ to a constant value.
Initialize states sequence $S = [s_1]$.
Initialize return-to-go sequence $G = [G_1]$.
repeat
  Generate action $a_t$ autoregressively using $\pi$ conditioned on $S$ and $G$: $a_t = \pi(S, G|\theta^\pi)$.
  Execute action $a_t$ and observe new state $s_{t+1}$.
  Append new state to states sequence: $S = [S; s_{t+1}]$.
  Update return-to-go $G_{t+1}$ using Equation 1.
  Append updated return-to-go to sequence: $G = [G; G_{t+1}]$.
until done
```

as SAC and outperforms all other baselines. This results demonstrate the effectiveness of our method in enhancing the actor-critic methods by introducing sequential information into the network. For *Walker2d-v2*, it failed to outperform SAC mainly due to the inherent complexities and challenges present in the *Walker2d-v2* environment. Unlike *Ant-v2* and *HalfCheetah-v2*, where SACFormer's enhanced capability to process sequential information proves significantly advantageous, the *Walker2d-v2* environment presents unique dynamics that are less amenable to this approach.

The *Walker2d-v2* environment requires a more nuanced balance and coordination, aspects that are not solely dependent on the agent's ability to process sequential information. This indicates a potential area for further improvement in SACFormer. Specifically, it suggests the need for a more sophisticated approach to handle environments where dynamic balancing and intricate motor control play a critical role in achieving high performance.

4.4. Ablation Study

In this section, we explore how the length of sequence modeling during training affects the overall model performance. In this experiment, we implement our method to the *Humanoid-v2* environment since it has higher state and action dimensions compared with environments explored in Figure 2, which can be more sensitive to sequence information. We set the sequence length K to 5, 10, and 15 during training and run the experiment for 0.8 million steps. The results are depicted in Figure 3. The model achieves optimal performance at $K = 10$, which aligns with our expectations. A shorter sequence length may fail to encapsulate sufficient sequential information, thereby increasing model complexity unnecessarily. Conversely, a longer sequence length could lead to misinterpretations or result in hallucinated outputs, especially when processing more extended contexts. Thus, a moderate length $K = 10$ strikes a balance, effectively capturing relevant information without the drawbacks associated with shorter or longer sequence lengths. Note that the best setting of sequence length can vary between environments. Therefore, different environments can have different optimal K values.

5. Conclusion

In this project, we present SACFomer, an online RL framework based on DT and SAC to enhance actor-critic methods with sequence modeling and enable online RL in the architecture of DT. The method redesigns the actor network with GPT-2 to model sequential relationships instead of normal deep neural networks. Additionally, it adheres to SAC's loss definition, establishing a convergence criterion for the model. The experiments demonstrated that our method attains competitive performance in three out of four continuous control tasks. On top of that, an ablation study on sequence length has also demonstrated the effectiveness

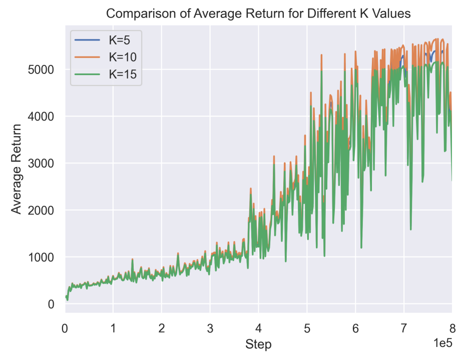


Figure 3. Ablation experiment on sequence length K. The results indicate that model with a moderate sequence length has the best performance.

of the sequence information and how sequence length can affect model performance. However, our method exhibits certain limitations, such as slower convergence compared to conventional Deep Q-Learning methods, primarily due to the extended time required for the transformer to develop an adequate feature representation. Additionally, the learning process is not as stable as standard neural networks since the GPT-2 architecture is deeper and more complex, and thus tends to output higher variance. We believe in the future, it is worth working on addressing SACFomer's current limitations and expanding research on online RL within the DT framework.

Acknowledgements

We thank our esteemed professors for their enlightening lectures and to our teaching assistants for their invaluable support throughout this course. We extend our heartfelt thanks to Tianyi Xu for generously providing his workstation for our experimental needs.

Contributions

We acknowledge the significant contributions of our team members to the project: Yuan Qing for his pivotal role in idea formulation, framework design, and experimental execution; Tianyi Xu for his thorough research on related works and setting up the experimental environment; and Yida Wang for his dedicated research on related works.

References

Achiam, J. Spinning Up in Deep Reinforcement Learning. 2018.

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay, 2018.

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. R. Rewriting history with inverse rl: Hindsight inference for policy improvement. *Advances in neural information processing systems*, 33:14783–14795, 2020.

Floridi, L. and Chiriatti, M. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018. URL http://arxiv.org/abs/1802.09477.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1273–1286. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *CoRR*, abs/1906.00949, 2019. URL http://arxiv.org/abs/1906.00949.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *CoRR*, abs/2006.04779, 2020. URL https://arxiv.org/abs/2006.04779.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35: 27921–27936, 2022.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020. URL https://arxiv.org/abs/2005.01643.

Li, Q. Continuous control benchmark of deepmind control suite and mujoco. https://github.com/LQNew/Continuous_Control_Benchmark, 2021.

Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z., and Ye, D. A survey on transformers in reinforcement learning. *arXiv preprint arXiv:2301.03044*, 2023.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015a.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N. M. O., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015b. URL https://api.semanticscholar.org/CorpusID:16326763.

Liu, F., Viano, L., and Cevher, V. Provable benefits of general coverage conditions in efficient online rl with function approximation. *arXiv preprint arXiv:2304.12886*, 2023.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

Meng, L., Wen, M., Yang, Y., Le, C., Li, X., Zhang, W., Wen, Y., Zhang, H., Wang, J., and Xu, B. Offline pretrained multi-agent decision transformer: One big sequence model tackles all smac tasks. *arXiv preprint arXiv:2112.02845*, 2021.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL https://api.semanticscholar.org/CorpusID:205242740.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations, 2018.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Radford, A. and Narasimhan, K. Improving language understanding by generative pre-training. 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention, 2021.

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

Wallbridge, J. Transformers for limit order books, 2020.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. Cvt: Introducing convolutions to vision transformers, 2021.

Xie, Z., Lin, Z., Li, J., Li, S., and Ye, D. Pretraining in deep reinforcement learning: A survey. *arXiv preprint arXiv:2211.03959*, 2022.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pp. 24631–24645. PMLR, 2022.

Zhao, K., Ma, Y., Liu, J., Zheng, Y., and Meng, Z. Ensemble-based offline-to-online reinforcement learning: From pessimistic learning to optimistic exploration. *arXiv preprint arXiv:2306.06871*, 2023.

Ziebart, B. D. Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. 12 2010. doi: 10.1184/R1/6720692.v1. URL https://kilthub.cmu.edu/articles/thesis/Modeling_Purposeful_Adaptive_Behavior_with_the_Principle_of_Maximum_Causal_Entropy/6720692.