Decision Transformer for Robot Imitation Learning

Alex Chandler
alex.chandler@utexas.edu
University of Texas at Austin

Jake Grigsby
grigsby@cs.utexas.edu
University of Texas at Austin

Omeed Tehrani
omeed@cs.utexas.edu
University of Texas at Austin

***Abstract*—Reinforcement learning typically involves an agent interacting with an environment to achieve a maximum reward. Our project disregards the traditional approach of estimating policies and simplifies Reinforcement Learning to a sequence modeling problem that can effectively be solved by the Transformer architecture. Our project extends the capabilities of the initial Decision Transformer (DT) [4] to learn from mixed-quality input data. Our modified Decision Transformer quantifies the benefit of return-conditioned imitation learning on mixed-quality data by leveraging the robomimic datasets. We show that our Decision Transformer significantly outperforms standard behavioral cloning on mixed-quality data for the Lift and Can tasks. Overall, our Decision Transformer and semi-sparse reward function provide a new way to tackle the challenges of imitation learning with mixed-quality data.**

I. Introduction

Robot imitation learning methods use supervised learning to mimic the actions necessary to complete complex tasks. This process typically involves a time-consuming cycle of collecting and filtering high-quality human demonstrations. In order to increase data availability and ease of use, we would like to learn from sub-optimal demonstrations. Sources of sub-optimal data may include inexperienced human teleoperators, scripted agents, or past versions of our own learning-based policy. Learning from fixed datasets of mixed-quality experience is often studied in offline Reinforcement Learning (RL) [17]. Despite a recent surge in research interest, offline RL remains challenging in practice, and it can be difficult to develop a method that achieves real-world results. Our project focuses on the Decision Transformer [4] - a simple and powerful alternative to RL algorithms that replaces brittle dynamic programming and policy gradients with standard sequence modeling. Like more traditional offline RL methods, Decision Transformer (DT) can learn from mixed-quality demonstrations by conditioning action outputs on the total return of each trajectory. DT policies can then imitate high-quality data while gathering extra information about the task by modeling low-quality demonstrations. At test-time, we can specify an expert-level target return and deploy a policy with the performance of expert imitation learning without the need for expensive curated datasets. DT reduces offline RL to a training routine that resembles sequence models in supervised learning. There is optimism that we can leverage diverse datasets and large network architectures similar to those that have been successful in domains like natural language processing to improve generalization in robot decision-making.

In this project, we implement a custom version of the Decision Transformer focusing on continuous action spaces and stochastic policies. We evaluate our implementations' ability to learn from mixed-quality data using two simulated domains from the robomimic benchmark [19] that provide machine-generated and expert demonstrations. We study the impact of several design decisions on task success rates, including model size, policy parameterization, and context sequence lengths. Results demonstrate that our DT implementation significantly outperforms naive Behavioral Cloning in this mixed-quality and multi-modal setting. We also discuss some limitations of our work and opportunities for future improvement on the robomimic benchmark. The code for this project is open-sourced and available on GitHub to enable future research[1].

II. Background and Related Work

A. Reinforcement Learning

In the Reinforcement Learning setting considered in this paper, an agent interacts with an environment in order to maximize a reward signal. At each timestep t, the agent perceives a state s_t and selects an action a_t, leading to a new state s_{t+1} and a reward r_t. The agent's decision-making strategy is defined by its policy π, which maps states to a distribution over actions $\pi(a_t \mid s_t)$. Our goal at every timestep is to maximize the cumulative sum of rewards until some time limit of H steps, also known as the return. We will use the term "return-to-go" (RTG) to refer to the return after a specific timestep t. The RTG at timestep t is denoted $\hat{R}_t = \sum_{i=t}^{H} r_i$. Decision Transformer computes RTG values in hindsight from an offline dataset and uses them to differentiate between low and high-quality demonstrations.

B. Offline Reinforcement Learning

Offline RL [17] studies decision-making from fixed datasets of prior experience. Unlike standard "online" RL, offline agents cannot explore the environment to gather new information about the task. Instead, offline methods attempt to stitch together optimal decision-making strategies from datasets of sub-optimal data to match or exceed the performance of the policies that generated the training dataset. Offline algorithms often have to confront technical challenges arising from situations our dataset does not cover. For example, methods such as CQL [15], CRR [28], and BCQ [10] restrict their

[1]https://github.com/jakegrigsby/robomimic-decision-transformer

policies' action choices to those present in the dataset in order to avoid out-of-distribution optimization issues. Model-based methods such as MOPO [31] explicitly penalize actions that lead to uncertain states. The ultimate goal of offline RL is to develop a scalable way to reuse data and deploy RL to real-world tasks, where data collection is expensive or unsafe. Examples include self-driving cars, healthcare, and real-world robotics [17]. However, current methods can be difficult to develop and tune without the ability to evaluate policies in the test environment. Recent research has worked to simplify techniques [9] and develop a repeatable workflow [16] for new applications.

C. Robomimic

Robomimic[2] [19] is a framework for offline robot learning from demonstrations as a part of the Advancing Robot Intelligence through Simulated Environments (ARISE) Initiative. Robomimic includes a unique collection of standardized human-collected and machine-generated datasets for offline RL and imitation learning. The benchmark currently supports several simulated manipulation tasks including "Lift", "Can", "Tool Hang", and "Square Transport". These tasks use the robosuite simulator [33] to enable affordable access to offline robotics research. These simulated environments can be configured to use low-dimensional proprioceptive states instead of high-dimensional rendered images. Robomimic also includes real-robot variants of similar tasks with camera-based observations.

Robomimic divides each task's offline dataset into several categories based on demonstration quality and extensively benchmarks behavioral cloning (BC) and offline RL algorithms. The results suggest that learning from a mixture of human datasets can be challenging because of the varying proficiency and strategies of multiple demonstrators. Each teleoperator solves the same task slightly differently, which leads to multi-modality even in simple tasks; this effect would likely be more problematic in the kinds of long-horizon tasks we would eventually like robots to solve. The robomimic authors find that offline RL methods can be challenging to use in practice and are often outperformed by simpler BC techniques. In particular, they find that the strongest baseline is a memory-equipped BC policy that uses a Recurrent Neural Network (RNN) to process sequences of observations. This BC-RNN baseline improves action prediction by using its memory of past timesteps to address the multi-modality of mixed human demonstrators. However, BC-RNN still struggles when the dataset contains low-quality demonstrations because it learns to imitate all of the decisions in the training set as one unified policy.

D. Outcome-Conditioned Behavioral Cloning

Recent work has looked to simplify offline RL by creating similarities to supervised learning techniques in related fields like natural language processing and computer vision. One promising approach adds the return-to-go of the offline trajectory as an extra input to a BC algorithm: $\pi(a_t \mid s_t, \hat{R}_t)$. This lets the policy differentiate between high and low-quality experiences during training. Low-quality data helps increase the size of the training set and may provide some valuable knowledge of the task. However, we can replicate only the high-quality actions at test-time by specifying an expert-level target return $\hat{R}_0$. Return-conditioned behavior cloning can often match or outperform more complicated RL algorithms [7, 14, 2]. The idea of conditioning behavioral cloning on the quality of the demonstration datasets also appears as "Upside Down RL" [23, 26]. The method has conceptual similarities to goal-conditioned reinforcement learning [20, 1], which we can reformulate as a supervised learning problem conditioned on states instead of returns [8, 11].

E. Decision Transformers

The Transformer architecture [27] uses an attention mechanism to process sequences without convolution or recurrence. Originally utilized for machine translation in natural language processing, the Transformer architecture has since appeared in state-of-the-art models for tasks across nearly every domain in machine learning. Transformers rely on a scaled dot-product attention mechanism that enables the representations of each timestep to interpret patterns across all the preceding timesteps in the sequence. Transformers are highly compatible with GPU parallelization and can learn complex temporal dependencies. Transformers are generally known to outperform RNN models while being more computationally efficient.

Decision Transformer [4] is an algorithm that combines the simplicity of return-conditioned BC with the performance of large Transformer networks. The Trajectory Transformer [13] is a concurrent work that adds language modeling techniques like beam search to RL. Additional extensions of the Decision Transformer concept include online fine-tuning [32], few-shot prompting [30], and applications to stochastic environments [21]. Finally, Gato [22] uses a similar architecture and tokenization scheme to Decision Transformer, although on a much larger scale and without return-conditioning.

III. Method

A. Data

Our data is collected and processed from the robomimic dataset [19], providing us with a large-scale dataset of robot demonstrations for a variety of tasks. The dataset includes three types of data: machine-generated (MG), proficient-human (PH), and multi-human (MH). The machine-generated data is generated by state-of-the-art reinforcement learning agents and is of varying quality, ranging from sub-optimal to high-quality. The proficient-human and multi-human data are demonstrations from teleoperators with varying levels of proficiency.

In this work, we focus on two tasks from the robomimic dataset: Lift and Can. The Lift task involves a robot lifting a cube, while the Can task involves picking up a can and placing it in the correct bin. These tasks were chosen because

[2]https://github.com/ARISE-Initiative/robomimic

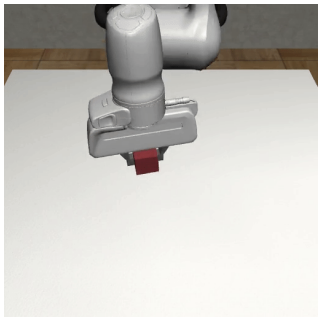

Fig. 1. The Lift task, where the goal is to pick a red cube off of the tabletop.

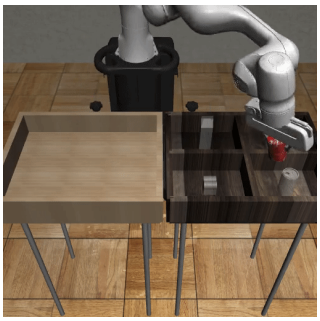

Fig. 2. The Can task, where the goal is to pick up the red can and deposit it in the proper container.

robomimic provides a large number of machine-generated (low-quality) demonstrations for those two tasks, which highlights the problem of learning from mixed datasets. To evaluate our approach, we create a fourth dataset for each task. This "All" dataset category is a challenging combination of the machine-generated, multi-human, and proficient-human data. Because there is much more MG data than human demonstrations, the "All" dataset is weighted towards lower-quality data, making it an interesting challenge for our purposes. Using this dataset, we can test DT's ability to learn from both high-quality and low-quality demonstrations. The Lift environment is shown in Figure 1 and the Can environment is shown in Figure 2.

B. Semi-Sparse Reward Function

The offline datasets in the original robomimic study use a binary reward function where the agent receives a reward of 1 for successful completion of the task and 0 at all other timesteps. Decision Transformer uses dense reward functions to differentiate between many different modes of policy quality [4]. Therefore, our first step was to assign a more dense reward signal to robomimic's manipulation tasks. The robomimic codebase does provide an option to relabel datasets with robosuite's dense reward function. However, we found that the resulting rewards were not correlated with the quality of the dataset. In the Lift task dataset, for example, the proficient-human demos had the highest success rate but the lowest return values, while machine-generated data had the lowest success rate but the highest returns. This was a result of the worse quality demonstrations compensating for their low reward per timestep by taking much longer to solve the task, leading to a higher total return. To address the issue, we manually changed the reward function to include a semi-sparse success bonus. We added a large positive reward upon completion of the task which decreases with every timestep. Let $r_t^{\text{robomimic}}$ be the original dense reward term provided by the robomimic dataset at time t, and d_t be a binary "done" flag where $d_t = 1$ indicates successful task completion. We define a new reward term r_t^{DT}:

$$r_t^{\text{DT}} = r_t^{\text{robomimic}} + d_t \max(500 - t, 0) \quad (1)$$

The success bonus decreases to zero after 500 timesteps, which is the maximum length of the robomimic tasks. We relabel the offline dataset using Equation 1, and modify the reward function in the environment simulator for evaluation. This design lets our DT learn from a wide range of target values and improve its understanding of the quality of the sequence data, even in the presence of low-quality data.

C. Decision Transformer Architecture

Our implementation is a modification of the original Decision Transformer architecture. The input format is a *context sequence* of up to k *tokens*, where k is the context length. Each token is a vector that concatenates a state s_t, return-to-go $\hat{R}_t$, and the previous action a_{t-1}[3]. All sequences are padded to a length of k. The resulting sequence is projected to the embedding dimension of the Transformer architecture with a two-layer feedforward network. Transformers are permutation-invariant and require a position encoding to interpret the order of input tokens. The original Transformer [27] used a hard-coded sinusoidal embedding. Instead, we randomly initialize embedding vectors for each of the k position indices and train them alongside the rest of the model. This solution is more popular in domains with continuous inputs such as Vision Transformers [6]. The $(s, a, \hat{R})$ and position embeddings are summed together and form the input sequence of the core Transformer architecture. Our input format is a slight departure from the original DT, which embedded states, actions, and RTGs as separate consecutive tokens. Our concatenated version simplifies the implementation and position embedding while saving compute by reducing the overall sequence length by a factor of 3.

The embedded sequence of tokens is then fed into a causal Transformer, which uses a self-attention mechanism to model the dependencies between the tokens. We use a Pre-Norm Transformer [29] architecture, which shifts the location of the layer normalization component to the residual branch. This is thought to improve stability early in training and enable deeper networks. The output of the Transformer is a sequence with the same dimensions as the embedded input. After a final normalization layer, this sequence is projected by a two-layer feedforward network into the dimension of the action distribution. The original DT uses a deterministic policy that outputs the action vector directly and is trained with mean squared error. We are focused on manipulation tasks with highly multi-modal demonstrations that would be difficult to model with a deterministic policy. Instead, our network generates the parameters of a stochastic distribution over actions. We implement a Gaussian policy with independent parameters for each dimension of the action space - a common default in the RL literature (e.g., [24, 12]). We also add a Gaussian Mixture Model (GMM) policy based on robomimic that adds extra parameters in order to better represent multi-modal action distributions. We use the GMM parameterization

[3]The first timestep of every trajectory is missing a previous action (a_{-1}), which we define as a zero vector.

with 5 modes by default, and compare the two policy types in the Section IV. A high-level summary of our DT model is depicted in Figure 3.

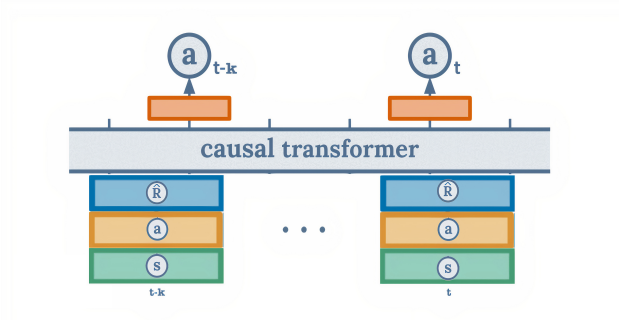

Fig. 3. Our Decision Transformer architecture, with states, actions, and return-to-go values concatenated into one token. DT maps a context sequence of k tokens to a sequence of k action distributions.

D. Training

Decision Transformer uses a standard sequence modeling objective and training loop. This can greatly simplify its implementation relative to many techniques in offline RL, where best practices are an open research topic [16]. We optimize our network parameters θ to maximize the probability of true actions in the dataset ($\mathcal{D}$) given the sequence of k previous states, actions, and RTG values:

$$c_t := (s_t, a_{t-1}, \hat{R}_t)$$
$$\theta^* := \underset{a_t, c_t, \ldots, c_{t-k} \sim \mathcal{D}}{E} [-\log \pi_\theta(a_t \mid c_t, \ldots, c_{t-k})] \quad (2)$$

We use several empirical best practices from the wider Transformer literature to stabilize training, including the AdamW optimizer [18] and a linear learning rate warm-up for the first two thousand gradient steps [27]. The batch size is fixed at 256. Training is performed on a single NVIDIA GeForce RTX-3090 and takes roughly 6 hours depending on model size. Much of the training time is consumed by evaluation rollouts in the simulated environments which are comparatively slow. We perform the evaluation in parallel across 8 independent environments, batching and padding the sequences into one forward pass of the Transformer.

IV. Experiments

In this section, we explore the challenges of action and return conditioned imitation learning and provide results for the Can and Lift tasks. We investigate the effects of introducing inaccuracies at test-time by conditioning the policy on its own slightly inaccurate history of actions and return-to-go values. Our experiments show that these inaccuracies can lead to lower task success rates for Decision Transformers (DT) compared to behavior cloning (BC) baselines. We then train DT models with various context sequence lengths on the more difficult "All" data setting of the Can task and show that longer context sequences improve prediction accuracy. Finally, we compare DT against BC baselines on the mixed-quality

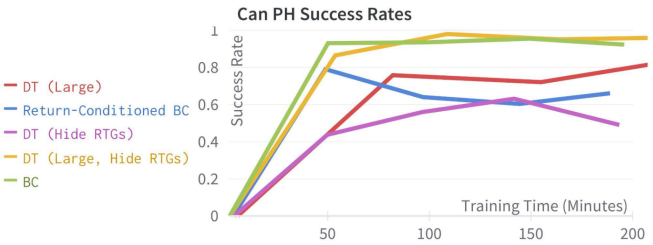

Fig. 4. Success rate of ablations of the DT input format in the Can task using only the highest quality PH data. "Large" policies double the networks' embedding and feedforward dimensions.

"All" dataset, demonstrating the effectiveness of DT's return-conditioning and context sequences for imitation learning in complex environments.

A. Understanding the Challenges of Action and Return-Conditioned Imitation Learning

Decision Transformers condition their outputs on a fixed-length history of previous actions, return-to-go scalars, and observations. While this may help differentiate between the various demonstrators in a mixed-quality dataset, it also introduces new opportunities for distributional shifts during evaluation: at test-time, the policy is conditioned on the history of its own (slightly inaccurate) actions and return-to-go values. These inaccuracies can compound over the length of a trajectory, potentially causing DT to achieve a lower task success rate than a BC baseline. As a first step, we investigate this effect on the Can placement task using only the highest quality dataset of human demonstrations (PH). In this setting, DT's ability to learn from mixed-quality data is less relevant. We use context lengths of one to compare directly against BC, meaning that the input to a DT model during evaluation is the most recent state, action, and return-to-go. We can optionally hide the action and/or RTG information. Hiding actions and RTGs recover the simple BC baseline, and a context length of one makes the attention component of the Transformer redundant.

Success rates are measured by pausing training and evaluating the current policy in the robomimic environment simulator. The success rates of several ablations of DT input format over the course of training are plotted in Figure 4. While standard BC matches the performance of the demonstrations and the original robomimic baselines, DT variants that introduce action and RTG inputs can struggle to achieve expert results. Increasing model size appears to improve performance by outputting more accurate actions and being more robust to variations in the input.

B. Decision Transformer on Can and Lift Tasks

Although action and RTG inputs may harm performance at test-time, our next experiment investigates whether DT's sequence inputs improve the accuracy of action predictions during training. We train identical DT agents with context lengths of $3, 10$ and 20 on the more difficult "All" data

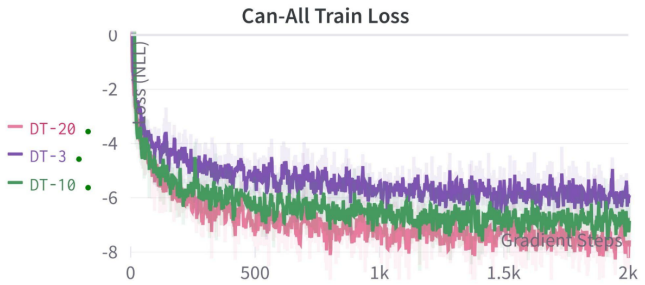

Fig. 5. Training Loss of DT with context lengths of $3, 10$ and 20. Longer context sequences improve action predictions by resolving ambiguity about which demonstrator is being imitated.

setting of the Can task. We would expect longer context sequence lengths to improve prediction accuracy by providing more information on the quality and action choices of the demonstrator that created each training sequence. The negative log-likelihood training loss (Equation 2) for the three models is plotted in Figure 5. Despite equal model sizes, regularization, and optimization settings, DT's fit of the training data improves with context sequence length.

Next, we conduct a larger-scale evaluation of DT with various context lengths and policy types. We compare against simpler BC baselines on the mixed-quality "All" dataset types consisting of both low-quality machine-generated data and high-quality human demonstrations. The results for the Lift task are shown in Table I, where DT-k denotes our Decision Transformer model with a context sequence of length k. Because DT is conditioned on the return-to-go values of its training data, we need to select a target return to imitate at test-time. We settle on the simple heuristic of finding the 95th percentile of the returns in the training dataset. We explore this choice further in Section IV-D. The *Naive BC* method uses a Gaussian policy without context sequences or return-conditioning and achieves a success rate of just 35% - highlighting the challenges of imitation learning from mixed-quality datasets with multiple demonstrators. For comparison, the original robomimic BC results achieve a success rate of 100% using the PH data and 65% from the lowest-quality MG data. We replicate these experiments with our codebase and report success rates of 99% for *Naive BC* on PH data. This suggests that learning from the mixed "All" dataset is more difficult than learning from any of its sub-components as a result of multi-modality. DT uses return-conditioning to filter dataset quality and context sequences to resolve multi-modality. As a sanity check, we train DT-1 on the PH data with 100% success. DT-1 still sees a decrease in performance on the "All" dataset but uses its return-conditioning to improve significantly on the BC baseline with a success rate of 85%. By extending the context sequence to 20 timesteps, DT-20 can further improve to 94%.

Table II shows the results of a similar experiment on the Can task. Although we successfully replicate the near-perfect success rate of the original robomimic results on the PH data, Naive BC once again fails (14%) on our mixed dataset. The relative performance of the methods on Can is similar to Lift, although we find the return and action conditioning to have a negative impact on this task as noted in Section IV-A. DT-3 achieves the highest performance on Can-All with a task success rate of 81%. Further discussion of the challenges in recovering the 100% success rate of the PH dataset with our Decision Transformer model is provided in Section V.

C. Ablations

There are two key implementation details that affect the performance of the Decision Transformer which could benefit from further analysis. First is the size of the Transformer architecture. While massive multi-billion parameter Transformers have been essential for progress in language modeling [3] when learning from internet-scale datasets, recent applications of Transformers to RL and robotic control problems have preferred much smaller model sizes [25, 4, 5]. We compare the performance of the larger Transformer used in the main Can-All experiments (Table II) - which used an embedding dimension of 512, a feedforward dimension of 2048, and 4 Transformer layers - with a smaller architecture of 3 layers with embedding dimension 200 and feedforward dimension 800. The smaller architecture is the same as was used for the easier Lift-All experiments (Table I). The results are shown in Table III. The larger model appears to provide a meaningful improvement in performance across all three context lengths. Because expert human demonstrations are expensive to collect, related work in Transformers for robotic imitation learning uses architectures much smaller than even our "Small" model to prevent overfitting. DT's return-conditioning offers a promising way to make use of diverse experiences and benefit from the generalization of large networks.

Next, we compare two choices of policy parameterization. The independent Gaussian policy is widely used in the RL literature for continuous control. However, the multi-modal policies created by a mixture of human demonstrations may benefit from a more expressive policy. We include a Gaussian Mixture Model (GMM) policy and enable it by default in all previous experiments with the exception of Naive BC. In Table IV, we directly compare the two policy choices at three context lengths on the Lift-All task, where GMM outperforms the Gaussian alternative in all cases. In general, we would expect the gap between the two policy choices to decrease as the context length grows and the action sequence becomes more specific to a single demonstrator and therefore less multi-modal.

D. Return-Conditioning

Because Decision Transformer models the actions of a range of policies of different quality, we are forced to select a "target return" or initial RTG that creates the first input token of the context sequence. While the target return can be difficult to select in an online setting [32, 26], in our offline case we can make an informed decision by looking at the distribution of returns available to us in the dataset. All results so far

TABLE I
LIFT-ALL RESULTS

	Naive BC	BC + Action Inp.	DT-1 (PH Only)	DT-1	DT-3	DT-10	DT-20
Success Rate (%)	35	20	*100*	85	92	92	**94**
Return	189	113	*463*	397	428	**433**	421

TABLE II
CAN-ALL RESULTS

	BC (PH Only)	Naive BC	BC + Action Inp.	DT-3	DT-10	DT-20
Success Rate (%)	*99*	14	15	**81**	76	63
Return	*396*	72	74	**362**	337	286

TABLE III
DT NETWORK SIZES ON CAN-ALL

	DT-3 (Large)	DT-10 (Large)	DT-20 (Large)	DT-3 (Small)	DT-10 (Small)	DT-20 (Small)	DT-3 (Large, Gaussian)
Success Rate (%)	**81**	76	63	65	57	61	66
Return	**362**	337	286	292	262	278	204

TABLE IV
GMM VS. GAUSSIAN ON LIFT-ALL

	DT-3	DT-10	DT-20	DT-3 (Gaussian)	DT-10 (Gaussian)	DT-20 (Gaussian)
Success Rate (%)	92	92	**94**	85	86	81
Return	428	**433**	421	403	406	386

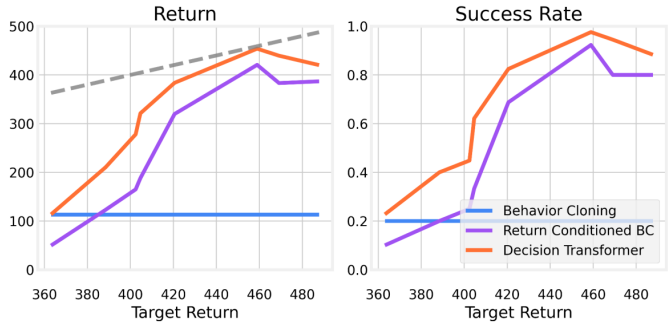

Fig. 6. Return-conditioned policies replicate a range of performance qualities. As the target return increases, so does the actual return and success rate of the agents.

have used the heuristic of replicating the 95th percentile of the returns in the training data. This decision was motivated by our desire to replicate high-quality behavior while staying safely inside the distribution of the training data. However, we can experiment with other choices of initial RTG. We evaluate trained policies across a range of target returns and plot the results in Figure 6. Because standard BC is not return-conditioned, the target return has no effect and the policy performs equally poorly in all conditions. Adding the target return as an extra input to BC does allow for adaptation to multiple levels of quality; as our target return increases so does the actual return (Fig. 6 left) and success rate (Fig. 6 right). Decision Transformer uses its context sequence to improve action predictions and more closely matches the target return than return-conditioned BC. Note that all policies significantly underperform the target return at the lowest quality levels ($360 - 400$). We speculate that this discrepancy is caused by difficulty in accurately mimicking low-return policies as a result of the way our custom reward function is correlated with demonstration length. It may be challenging to replicate slow solutions to the task with a limited context sequence and without time information in the state representation.

V. DISCUSSION AND FUTURE WORK

Our experiments suggest that return-conditioned imitation learning methods like Decision Transformers can outperform standard Behavioral Cloning when learning from mixed-quality data. However, our implementation of DT still underperforms BC models that use the highest quality subset of the dataset (PH). The real promise of Decision Transformer is its ability to mimic high-quality data while still learning whatever may be useful from a large quantity of sub-optimal demonstrations. Ideally, DT would strictly outperform the BC-PH baselines and make use of MG datasets in more difficult robomimic tasks like Square and Tool Hang. We offer three possible explanations for why we are not seeing these results, and suggest directions for future improvement.

First, limited computational resources have restricted our ability to perform large hyperparameter sweeps. It is possible that there are higher-performance model architectures, regularization settings, and other empirical details that could improve our results. We hope that our open-source code release will be a useful starting point for future experiments.

Second, there may be alternative reward functions that are more suitable for return-conditioned BC than the time-based completion bonus used in our experiments (Eq. 1). Rewarding

the agent for the time to completion clearly separates poor MG demonstrations from quality PH ones. However, it may over-emphasize fast solutions instead of consistent and easily repeatable behavior. Furthermore, the time-based rewards may be too difficult for the sequence model to imitate precisely because it is difficult to identify the current time with a fixed-length context window. Results such as Fig. 6 show that current policies struggle to imitate low-return (slow) demonstrations. We plan to add a timestep counter to the state representation in the open-source release and re-evaluate on the more difficult robomimic tasks.

Finally, there may be meaningful algorithmic changes needed to address some limitations of return-conditioned imitation learning in the robomimic setting. Because the robomimic datasets contain much more machine-generated data than human demonstrations, our return-to-go labeled dataset is heavily skewed toward low-return outcomes. Relabeling trajectories in hindsight with their achieved return creates an implicit "task-averaging" or return-agnostic term in the resulting policy, which can be especially problematic in cases where the outcome distribution is biased. For more on this issue, we refer the reader to [8].

VI. Conclusion

In this paper, we propose a modified version of the Decision Transformer for robotic imitation learning and evaluate its performance on sub-optimal datasets. We show that the Transformer's ability to understand context and long-term dependencies can improve action predictions and outperform return-conditioned behavioral cloning. However, we also find that conditioning on a history of previous actions and RTG scalars can hinder the model's performance and result in distributional shifts during evaluation. We demonstrate that increasing model size can improve performance on both tasks and that longer context sequence lengths can improve prediction accuracy during training. When compared to simpler baselines, we see clear improved performance on both tasks.

Our work has the potential to be applied in various robotics tasks, particularly those that involve working with sub-optimal or "noisy" data. General robotics research focuses on designing robots that can operate in unstructured environments, such as homes and offices, where data collection for manipulation tasks can be noisy and difficult. Our approach could potentially be a useful starting point for using Decision Transformers to improve the performance of robots in these types of settings. For instance, we can use past versions of data to improve our models over time, employ a wide range of policies on different data types, and reduce the need for extensive data cleaning before training. Additionally, future work should focus on mitigating the distributional shifts observed in our approach and exploring ways of improving the reward function to better support dense rewards. We open-source the entire project as a stepping stone in these endeavors.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] David Brandfonbrener, Alberto Bietti, Jacob Buckman, Romain Laroche, and Joan Bruna. When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv:2206.01079*, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33, 2020.

[4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021. URL https://arxiv.org/abs/2106.01345.

[5] Zichen Jeff Cui, Yibin Wang, Nur Muhammad, Lerrel Pinto, et al. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning?, 2021. URL https://arxiv.org/abs/2112.10751.

[8] Benjamin Eysenbach, Soumith Udatha, Sergey Levine, and Ruslan Salakhutdinov. Imitating past successes can be very suboptimal. *arXiv preprint arXiv:2206.03378*, 2022.

[9] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[10] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062. PMLR, 2019.

[11] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

[12] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint*

arXiv:1812.05905, 2018.

[13] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem, 2021. URL https://arxiv.org/abs/2106.02039.

[14] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies, 2019. URL https://arxiv.org/abs/1912.13465.

[15] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[16] Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021.

[17] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[19] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[20] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

[21] Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers fail in stochastic environments, 2022. URL https://arxiv.org/abs/2205.15967.

[22] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022. URL https://arxiv.org/abs/2205.06175.

[23] Juergen Schmidhuber. Reinforcement learning upside down: Don't predict rewards – just map them to actions, 2019. URL https://arxiv.org/abs/1912.02875.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[25] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022.

[26] Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning, 2019. URL https://arxiv.org/abs/1912.02877.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

[28] Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7768–7778. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/588cb956d6bbe67078f29f8de420a13d-Paper.pdf.

[29] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

[30] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *International Conference on Machine Learning*, pages 24631–24645. PMLR, 2022.

[31] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142, 2020.

[32] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer, 2022. URL https://arxiv.org/abs/2202.05607.

[33] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.