Evaluating Stitching Capabilities of RvS Transformer Algorithms

David Chen [1] **Arthi Suresh** [1] **Prateek Varshney** [1]

Abstract

Recent advancements in reinforcement learning (RL) have demonstrated that offline reinforcement learning can be cast as a sequence modeling problem, achieving competitive performance on several RL benchmarks. However, since Transformer-based Reinforcement Learning via Supervised Learning (RvS) algorithms do not explicitly maximize rewards, they typically struggle to stitch together suboptimal trajectories to generate optimal policies. While several improvements to the original Decision Transformer have been proposed, there is no comprehensive evaluation of these algorithms across difficult stitching environments. In this paper, we benchmark the stitching capabilities of various RvS methods on several AntMaze settings, including more challenging mazes omitted by most papers. We explore an improved waypoint selection mechanism for the Waypoint Transformer, and demonstrate performance improvements in difficult AntMaze settings. Additionally, we show that waypoints learned on one maze can be utilized to achieve better or comparable performance on another maze, highlighting the potential for transfer learning in goal-conditioned behavior cloning. Our contributions are two-fold: (1) We provide a more comprehensive comparison of the effective trajectory stitching capabilities of the different Transformer-based RvS methods, and (2) provide empirical results for our enhancements to current algorithms that suggest avenues for future research.

1. Introduction

In the classical reinforcement learning (RL) setting, an agent interacts with the environment to collect data and learn policies that maximize rewards. In settings where online interaction can be expensive or risky, offline RL, which learns policies from pre-collected data without online interaction, may be preferable. Recent work demonstrates that Transformer-based architectures enable recasting offline RL as a sequence modeling problem, yielding performance on tasks such as Atari, OpenAI Gym, and Key-to-Door that is on par with or exceeds other popular offline RL baselines such as conservative Q-learning and behavior cloning (Chen et al., 2021; Janner et al., 2021a; Bhargava et al., 2024). This reinforcement-learning-via-supervised-learning (RvS) paradigm presents the exciting opportunity to leverage the vast progress in the supervised learning and sequence learning domains for reinforcement learning problems.

The recent Transformer-based RvS algorithms draw from traditional behavior cloning methods but additionally condition on a desired outcome. Although promising, these outcome-conditioned behavior cloning methods do not explicitly maximize returns and suffer from an inability to stitch together segments of suboptimal trajectories, compared to dynamic programming methods. This limitation is amplified when training trajectories covering only portions of the path to the goal never fully reach the goal, which makes conditioning on the goal during these portions undefined. Figure 1. illustrates the stitching problem on `antmaze-large-play` from D4RL.

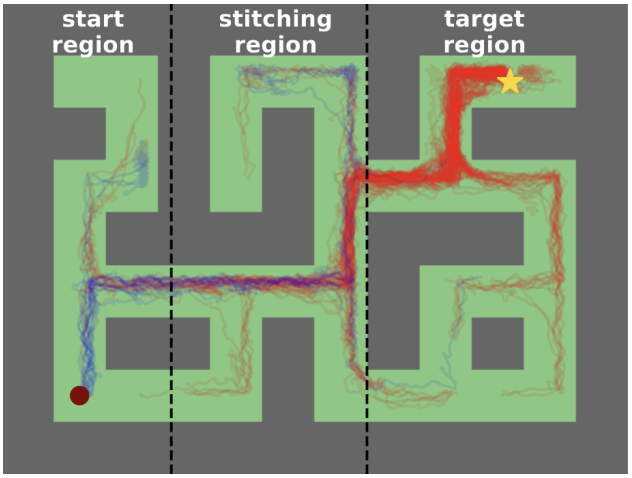


Figure 1. Borrowed from the Waypoint paper (Badrinath et al., 2023), the graphic represents training trajectories for `antmaze-large-play` navigating from the start location (circle) to the target location (star). Blue lines represent trajectories that pass through the starting region, while red lines represent trajectories that pass through the target region. Less than 5% of all trajectories pass through all regions.

In this paper, we address these challenges by first comprehensively benchmarking several improvements to the

original Decision Transformer, including the Decision ConvFormer, Elastic Decision Transformer (EDT), the Waypoint Transformer, and Reinformer. Our focus is on evaluating these methods in more challenging environments, such as `antmaze-medium` and `antmaze-large`, to provide a thorough assessment of their stitching capabilities. Additionally, we propose enhancements to the Waypoint Transformer (the current best performing method) by more effectively sampling waypoints, by weighting the MSE loss in the goal network by the returns-to-go, biasing the learned waypoints towards more "effective" waypoints rather than the average state K steps ahead, and dynamically choosing K. Lastly, we note that the choice of goal-conditioning as opposed to reward-conditioning seems to be consequential for the AntMaze settings. We attempt to further understand if learning waypoints using goal conditioning in one maze can achieve comparable performance when applied to another maze, demonstrating that the waypoint network imparts setting-agnostic navigation capabilities to the agent.

Summary of Contributions

1. We provide a comprehensive benchmarking of Transformer-based RvS algorithms on challenging antmaze environments.

2. We propose improved waypoint selection mechanisms for the Waypoint Transformer and provide empirical evidence that some of these directions may result in better performance.

3. We demonstrate the setting-agnostic navigation capabilities of the waypoint network and provide insights into the performance differences between goal-conditioning and reward-conditioning in the AntMaze settings.

2. Background

Recent advancements in offline RL have highlighted the potential of Transformer-based architectures to reframe RL as a sequence modeling problem. Notable works include the Decision Transformer, which leverages this paradigm to achieve competitive performance on a variety of tasks (Kim et al., 2023; Janner et al., 2021a; Bhargava et al., 2024). These methods may be preferred over traditional RL methods because they do not require explicitly learning any value function. However, these methods often struggle with stitching, the ability to combine segments of suboptimal trajectories to form a successful path to the goal.

2.1. Outcome-Conditioned RvS

The Decision Transformer and other related algorithms are part of a more general class of reinforcement learning algorithms that reduce the policy learning to a supervised learning problem. The distinguishing factor from traditional behavior cloning is the presence of a conditioning variable.

The two major types of conditioning in the reinforcement learning context are reward-conditioning and goal-conditioning. With reward-conditioning, we attempt to learn actions for trajectories that will give us a desired return. For goal-conditioning, we assume prior knowledge of a specific goal. We then learn actions for trajectories that lead us to the given goal.

As presented, these methods do not explicitly perform reward maximization.

2.2. Trajectory Stitching

Trajectory stitching in the context of offline reinforcement learning is necessary when training datasets contain several "suboptimal" trajectories, but some sequence of portions of the trajectories presents a valid and often optimal policy. This is analogous to having knowledge about how to get from a state A to state B, knowledge about how to get from state B to state C, but no direct experience or knowledge about how to get from state A to C.

This issue is easily overcome with traditional TD learning methods, since the Bellman backup present in these methods directly allows for seamlessly connecting similar states and their learned values across different trajectories. However, it is not trivial to address this issue with outcome-conditioned RvS methods. If the only trajectories available during one portion of the task are not conditioned on the final outcome, but are part of the actual required path in order to reach the final outcome, it is not obvious how the method would learn to associate the trajectories to the different desired outcome.

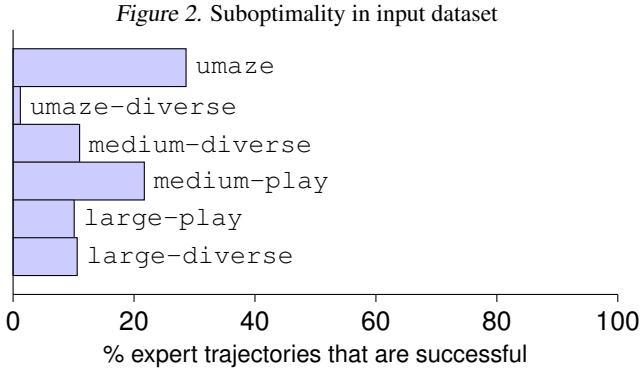
2.3. Current State of Stitching in RvS

Several improvements to the original Decision Transformer have been proposed which claim to improve upon stitching including the Decision ConvFormer, Elastic Decision Transformer (EDT), the Waypoint Transformer, and the Reinformer (Kim et al., 2023; Wu et al., 2023; Badrinath et al., 2023; Zhuang et al., 2024). EDT modifies the rewards used for conditioning by choosing a maximum over a number of context windows, recognizing that shorter windows ones may be better when the trajectory is suboptimal. The Waypoint Transformer generates intermediate targets or "waypoints" to help guide the model through stitching regions, and the Reinformer uses expectile regression to condition on estimated max returns achievable from a point, improving stitching by addressing the out-of-distribution issue with return conditioning.

Despite claiming to improve stitching, several of these papers only present results for `antmaze-umaze`, on which the original Decision Transformer already achieves compet-

itive performance. We propose that evaluating additionally on `antmaze-medium` and `antmaze-large` can provide a more thorough evaluation of an algorithm's stitching capabilities. Since these settings generally have a lower proportion of input trajectories that achieve the final goal (a metric we quantify in Figure 2) and a lower proportion of trajectories that cover all regions (see Figure 1), they allow us to assess the methods' robustness to learn good policies and stitch effectively under varying levels of suboptimality and overlap in the trajectories dataset. Our work extends

Figure 2. Suboptimality in input dataset



this evaluation to more complex environments, providing a clearer picture of the robustness and effectiveness of these algorithms in handling difficult stitching scenarios.

Other improvements that we chose not to evaluate include the use of a learned value function as a guide either during training or during inference (Wang et al., 2023; Janner et al., 2021b). There are also other methods that focus on solving the trajectory stitching issue by augmenting the data with learned transitions between trajectories (Li et al., 2024).

3. Approach

3.1. Evaluation of RvS Algorithms

We replicated the Decision Transformer (DT), Decision ConvFormer (DC), Elastic Decision Transformer (EDT), Waypoint Transformer (WT), and Reinformer based on the GitHub repositories associated with their respective papers. We reproduced and, where needed, extended evaluations to the AntMaze results, using hyperparameters provided in each of the individual papers.

Decision Transformer models trajectories autoregressively with returns-to-go model $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ as part of the input trajectory: $\tau = (\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \ldots, \hat{R}_T, s_T, a_T)$. At test time, we can specify the desired performance (e.g. 1 for success or 0 for failure), as well as the environment starting state, as the conditioning information to initiate generation. An embedding for each timestep is learned and added to each off the $3K$ tokens ($(s_t, a_t, \hat{R}_t)$ at K timestamps) which are then processed by a GPT model, to predict future actions using autoregressive modeling (See Figure 3).

While the original paper did not run experiments on the AntMaze problem, we use the same evaluation methodology as for other models. Since Decision ConvFormer only modifies the attention mechanism in DT, it uses the same evaluation/inference method as the Decision Transformer.

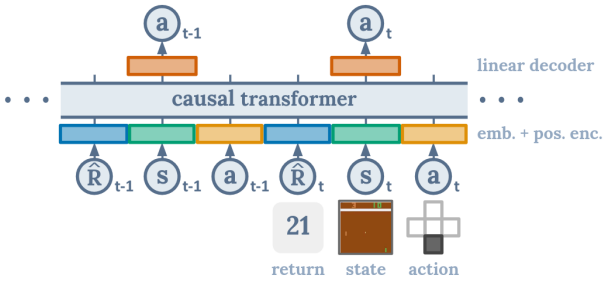


Figure 3. Decision Transformer applied to a sequence of states, actions, and returns-to-go, outputting actions.

The Elastic Decision Transformer (EDT) first estimates the maximum achievable return $\tilde{R}_i$ for each history length i and the maximal value provides the optimal history length w. Using w, it estimates the expert value $\tilde{R}_t$ at the corresponding truncated time step t using Bayes' Rules. Subsequently, the action to take is predicted using the causal transformer decoder by using the truncated traversed trajectory w.

The Reinformer takes a similar approach, but instead of modifying the context length, it aims to predict the maximum achievable return seen in the training trajectories at a given state. It then directly uses the predicted $\tilde{R}_i$ for conditioning.

All of the aforementioned algorithms tackle AntMaze as a reward-conditioned task. Waypoint Transformer presents a different approach using goal-conditioning. We describe more details on Waypoint Transformer and our improvements in the next section.

While the original papers for all of the mentioned methods are inconsistent in their evaluation, it likely does not change the conclusions regarding the gap between the Waypoint Transformer and other algorithms on the difficult mazes. For our evaluation, each algorithm, with the exception of Elastic Decision Transformer due to larger computational complexity and runtime, is evaluated on three different seeds. We report the mean and standard deviation of the D4RL score across these seeds. For AntMaze, the D4RL score can be interpreted as the fraction of rollouts that reach the goal.

3.2. Waypoint Transformer

The Waypoint Transformer leverages the proposed waypoint network W_ϕ and a GPT-2 architecture based on multi-head attention (See Figure 4). The WT policy π_θ is conditioned on past states $s_{t-k\ldots t}$ and waypoints (either generated goals

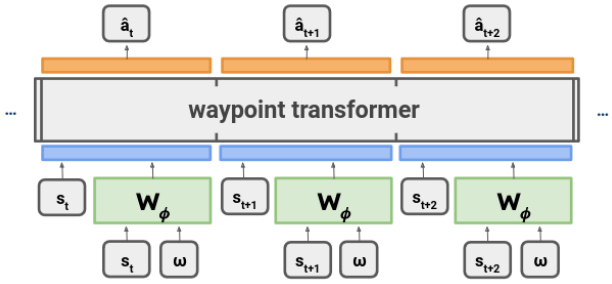


Figure 4. Waypoint Transformer architecture. $\Phi_t = W_\phi(s_t, \omega)$ represents the output of the goal/reward waypoint network.

or rewards), denoted as $\Phi_{t-k...t} = W_\phi(s_{t-k...t}, \omega)$.

To make the K-step predictions of future observations conditioned on the current state, s_t, and the target goal, ω, WT is trained to minimize the objective:

$$\arg\min_{\phi} \sum_{\tau \in D} L_\phi(W_\phi(s_t, \omega), s_{t+K}),$$

on the offline dataset D, where L_ϕ is the mean-squared error (MSE) loss for continuous state spaces. Further, unlike DT, the WT is not conditioned on past actions $a_{t-k...t}$; ϕ_t is concatenated with s_t to produce one token per timestep t, instead of multiple tokens used in DT.

3.3. Choosing Better Waypoints

The Waypoint Transformer trains the waypoint network by selecting random start and end indices in input trajectories, and predicting the location K steps ahead, where $K = 30$ was chosen after experimenting with various values. While this imparts the waypoint network with the general ability to generate intermediate targets given arbitrary start and end locations, there is nothing that meaningfully connects these targets to achieving the global goal during training.

To improve upon this, we explored a few hypotheses. First, we posit that waypoints associated with trajectories that achieve the global goal may be more useful to learn. While we feed in the global goal to the waypoint network during inference, we might benefit from the waypoint network attending more to the trajectories that are more relevant to achieving the global goal during training. We formulate the revised waypoint network task as the following, where $f(\hat{R}_t)$ is a monotonic function of the returns-to-go at time step t.

$$\arg\min_{\phi} \sum_{\tau \in \mathcal{D}} \mathcal{L}_\phi(W_\phi(s_t, \omega), s_{t+K}) \cdot f(\hat{R}_t) \qquad (1)$$

We experiment with a linear form for $f(\hat{R}_t) = b + c * \hat{R}_t$ where $b = 0.5, c = 10.0$.

In addition, we hypothesize that while $K = 30$ might have yielded the best empirical performance in experiments detailed in the Waypoint Transformer paper, different distances may be more optimal in different parts of the maze. For example, in areas with large divergence in trajectories, it might be useful to use waypoints that are closer to the current state as a way to "tread cautiously." Also, the original Waypoint paper noted that most of the trajectories that fail get stuck before the last turn, and we hypothesized that perhaps the state $K = 30$ steps ahead used for conditioning may overshoot the goal. Dynamically changing the distance closer to the goal may help in this case.

$$\arg\min_{\phi} \sum_{\tau \in \mathcal{D}} \mathcal{L}_\phi(W_\phi(s_t, \omega), [s_{t+i}, s_{t+2i}, ..., s_{t+K}]) \qquad (2)$$

$$\arg\min_{\phi} \sum_{\tau \in \mathcal{D}} -\log \pi_\theta(a_t | s_{t-k,..,t}, \Phi_{t-k,..,t}) \qquad (3)$$

First, we predict multiple waypoints $i, 2i, ..., K$ steps ahead, i.e. a total of $\lceil K/i \rceil$ waypoints, instead of just one waypoint K steps ahead. We choose $i = 5$ and $K = 30$ in our experiments. Then, we generate a weight for each waypoint, where this weighting module takes in the goal and the current observation. These weights are normalized using a softmax layer with high temperature and used to compute the weighted sum of the multiple waypoints via a dot product, to yield a single (x, y) which is a composition of the multiple waypoints. Because we use a temperature of 2, we are effectively choosing a single waypoint for conditioning. These weights are trained jointly with the transformer network.

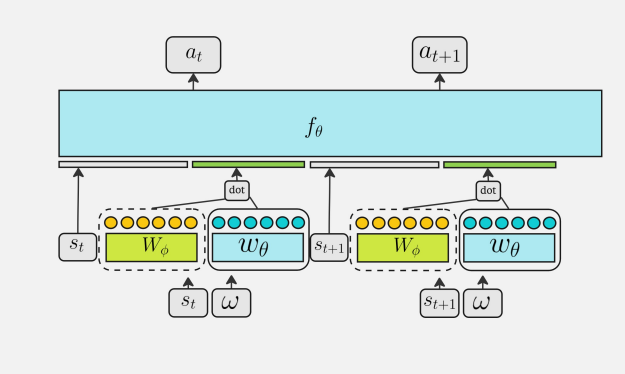


Figure 5. The waypoint network denoted by W_ϕ outputs n waypoints, and is trained independently. The transformer f_θ and the weights w_θ are learned together. The weights pass through a softmax and are dotted with the waypoints to produce a single (x, y) value to use for conditioning.

For these various "reweighting" schemes we focus our training and evaluation on `antmaze-large-diverse` and `antmaze-large-play` as these are representative settings for harder mazes on which the Waypoint Transformer

outperforms other methods, and where we might be able to drive performance even further.

Lastly, while we attempt to fine-tune waypoints to the goals of interest, we also explore how transferable the more general, global-goal-agnostic capabilities of the waypoint network are by training the waypoint goal network on `antmaze-large-diverse` and using this to train the transformer for a different setting, `antmaze-medium-diverse`.

4. Experiments and Results

4.1. Baseline Evaluation

Table 1 shows the performance of the aforementioned algorithms on these different settings. As expected, Decision Transformer, Elastic Decision Transformer, and the Decision ConvFormer perform very poorly on the more difficult AntMaze settings, while the Waypoint Transformer shows a success rate of above 60% for all of them. However, some of the other algorithms like Decision ConvFormer and Elastic Decision Transformer outperform Waypoint on `antmaze-umaze`. This validates our hypothesis that `antmaze-umaze` alone is not a good indicator of how the algorithms will fare in difficult stitching scenarios.

When we plot the trajectories taken by Elastic Decision Transformer and Waypoint Transformer to understand this performance gap, we see that the Elastic Decision Transformer can hardly make it past the first major turn in `large-play` (See Figure 6).

We are unable to reproduce results for Reinformer that show any meaningful improvement in the medium environments, and the performance in the large environments is also quite poor (see Table 2).

The performance of these algorithms on the `umaze` settings are expected based on the results in the original papers. Notably, Waypoint Transformer does not perform the best on `umaze`, and no algorithm demonstrates superior performance on all settings. This indicates an opportunity to do a more thorough analysis of where and why these algorithms fall short.

4.2. Impact of Goal-Conditioning

An understated but likely important difference between the Waypoint Transformer and all other methods is the choice of goal-conditioning as opposed to reward-conditioning. We attempted to explore this further by augmenting Reinformer by simply adding in information about achieved goals during training and desired goals during evaluation. This methodology is consistent with Waypoint Transformer as well as the "RvS-G" approach. We present additional results for a subset of the harder AntMaze environments, where "RvS-G" results can be considered as a simple non-waypoint but goal-conditioned RvS method. These results are pulled from the Waypoint Transformer paper.

According to the RvS-G results, the presence of an added goal should provide substantial improvement in `antmaze-large`, compared to not being able to solve the task at all. However, we were unable to produce any such improvement by simply augmenting Reinformer with goal-conditioning in addition to reward conditioning. The choice of hyperparameters and context length may have a huge impact in this case, as the original Reinformer paper required an extensive grid-search to select very specific hyperparameters for the presented results, including setting the context length to a mere 3 steps for AntMaze.

Table 2. Impact of goal-conditioning

Env.	RvS-G	Rein	Rein-G
`large-diverse`	36.9 ± 10.5	0.67 ± 1.2	0.0 ± 0.0
`large-play`	32.4 ± 10.5	2.3 ± 2.0	0.67 ± 1.2

4.3. Choosing Better Waypoints

Table 3. Revisiting waypoint selection

Env.	WT	Lin.	Mult. WP
`large-diverse`	68.7±8.3	**76.2± 2.4**	**76.8±6.0**
`large-play`	**73.2±2.0**	70.7± 4.0	68.8±6.3

While providing stronger performance on `large-diverse`, the reweighted loss schemes did not provide substantially and consistently better performance compared to the Waypoint Transformer. This may be because only 8% training instances have non-zero returns-to-go for the `large-diverse` setting, which means the reweighting, due to the sparsity, does not have much effect. Additionally, the values of b and c for the linear transformation of returns-to-go were arbitrarily chosen. Due to computational limitations, we did not perform a more exhaustive hyperparameter search, which may have yielded a configuration with better performance. Lastly, for the `large-diverse` setting, it is unclear if weighting examples should increase alignment to the goal given during rollouts, since the "diverse" dataset is generated by commanding random goal locations in the maze and navigating the ant to them.

Predicting multiple waypoints seems to have some marginal performance wins on `large-diverse`, but this is not replicated on `large-play`. When we tried to plot whether there was any differentiation at all in the distance of the waypoints chosen at different parts of the maze, we found

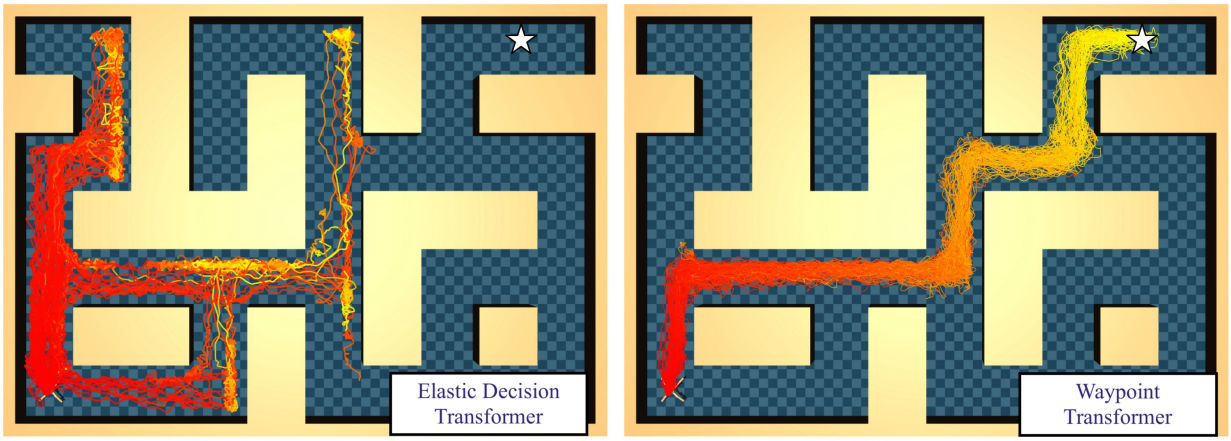


Figure 6. The left image shows 100 rollouts from Elastic Decision Transformer (EDT) on large-play-v2. Note how EDT is unable to stitch well enough to explore the right half of the maze. The right image shows 100 rollouts from WayPoint Transformer (WT) on large-play-v2. Note how WT does a great job at stitching and finds the optimal trajectory from the starting position to the goal.

Table 1. Results on AntMaze environments.

Env.	DT	DC	EDT	Reinformer	WT
`umaze`	67.5 ± 9.4	81.8 ± 0.8	66	**84.4 ± 2.7**	67.1±3.4
`umaze-diverse`	60.6± 12.0	87.5± 2.5	**91**	65.8 ± 4.1	65.3±8.8
`med-diverse`	1.25± 2.5	3.0± 1.2	2.0	7.3± 5.1	**61 ± 8.4**
`med-play`	5.6± 5.6	5.3± 3.6	0.0	3.3 ± 2.0	**64±6.4**
`large-diverse`	0.0± 0.0	1.3± 2.2	0.0	0.67 ± 1.2	**68.7±8.3**
`large-play`	0.63± 1.2	1.5± 2.1	0.0	2.3 ± 2.0	**73.2±2.0**

that no real patterns are observed; the model consistently picks the waypoint either 15 or 20 steps ahead, so it is possible our hypothesis that different waypoints are useful at different parts of the maze is not true. In Figure 7 we see that the waypoints generated appear to be smoother and more "on track" but according to our evaluation results this doesn't translate to better overall performance.

4.4. Transfer Learning

We previously stated that since we are not aligning or giving importance to global goals in the training of the waypoint network, there might be some opportunity to improve performance by doing so. However, the fact that the waypoint network provides more general, rather than goal-specific waypoints is interesting to explore in itself. When we train a waypoint goal network on `antmaze-large-diverse` and use it to train the transformer for a different maze, `antmaze-medium-diverse`, we find that it achieves comparable performance to training both networks on `antmaze-medium-diverse`, when evaluated on 3 seeds.

Table 4. Transfer learning between mazes

Transformer env.	Waypoint net env.	D4RL Score
`medium-diverse`	`medium-diverse`	61.6± 8.4
`medium-diverse`	`large-diverse`	**65.5± 3.5**

If we look at the waypoints generated by the waypoint network trained on `large-diverse` for the `medium-diverse` maze (See Figure 7), we see that they quite noisy, often predicting parts of the maze that are not reachable. Yet we observe that it has comparable or better D4RL score compared to training the waypoints on the same maze. This suggests that the accuracy and smoothness of the waypoints perhaps do not matter much at all, and also helps explain why our attempted modifications might not provide consistent improvements.

5. Conclusion

Most of the algorithms perform quite poorly on the more difficult AntMaze settings. This matches our hypothesis that despite improving stitching in some scenarios, these algorithms fail in harder environments. Any future algorithmic exploration that attempts to improve stitching should eval-

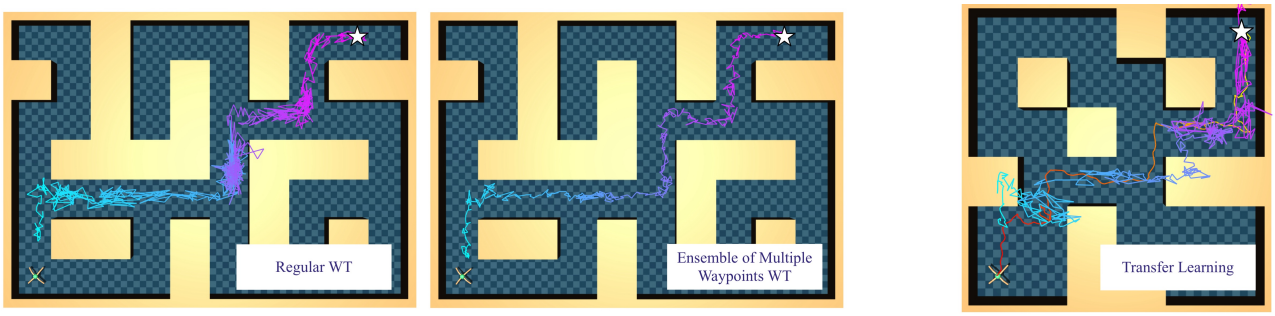


Figure 7. The first two images show the waypoints generated along a rollout for the regular Waypoint network and our modified version where the model selects from an ensemble of waypoints. The last image shows the waypoints generated by a transfer learning approach. The line that transitions from blue to purple shows the waypoints generated by the network trained on the larger maze setting, applied to the medium maze, and red-to-yellow line shows the actual trajectory.

uate on these settings to provide a comprehensive view of performance and motivate understanding of the deficiencies of existing methods.

Goal conditioning seems to be a crucial choice for higher performance, allowing algorithms to exploit spatial information in the more difficult AntMaze environments. Waypoint Transformer outperforms other methods on these harder mazes. Weighting the loss of the waypoint network by returns-to-go, and sampling from an ensemble of Waypoints demonstrate better performance on some but not all settings. After analyzing these in conjunction with the erratic waypoints generated by a transfer learning approach, which yields comparable-to-better performance, we conclude that the smoothness and accuracy of the waypoints may not matter much at all.

The fact that using waypoints generated by a model trained on one maze to learn a policy for another maze works considerably well presents interesting opportunities to learn a more general navigation model for goal-conditioned supervised learning. In particular, in low-data scenarios, we might be able to utilize data from similar settings to bootstrap the learning process and achieve better performance. This direction relates to "meta-learning" and leveraging pre-training on more general datasets to enable fine-tuning to a setting of interest. Future work can expand upon the results here and assess transferability across more varied settings.

6. Acknowledgements

We thank Anirudhan Badrinath, Garrett Thomas and Professor Emma Brunskill for providing mentorship and several code pointers for this project.

7. Code

All code for the attempted modifications can be found across the following repositories https://github.com/arthisuresh/waypoint-transformer, https://github.com/DavidJGChen/waypoint-transformer (adapted from https://github.com/StanfordAI4HI/waypoint-transformer. For other algorithms we reproduced we directly used the authors' Github code.

8. Team Contributions

Due to the nature of the project, we all worked together on some common aspects as well as the overall conceptual understanding of the work. For the baseline evaluations, David produced results for Decision Convformer, Reinformer, part of Waypoint Transformer, and Reinformer with additional goal information. Arthi produced results for Waypoint Transformer, Decision Transformer, and the attempted modifications to Waypoint (weighting, multiple waypoints, transfer learning). Prateek replicated results for Elastic Decision Transformer and developed the coding plotting code which was then modified and used to visualize trajectories of the different WT variants in collaboration with David. We analyzed the results together and brainstormed on various modifications as well as their intuition. All team members collaborated on the write-up.

References

Badrinath, A., Nie, A., Flet-Berliac, Y., and Brunskill, E. Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets, 2023.

Bhargava, P., Chitnis, R., Geramifard, A., Sodhani, S., and Zhang, A. When should we prefer decision transformers for offline reinforcement learning?, 2024.

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, 2021.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem, 2021a.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021b.

Kim, J., Lee, S., Kim, W., and Sung, Y. Decision convformer: Local filtering in metaformer is sufficient for decision making, 2023.

Li, G., Shan, Y., Zhu, Z., Long, T., and Zhang, W. Diffstitch: Boosting offline reinforcement learning with diffusion-based trajectory stitching, 2024.

Wang, Y., Yang, C., Wen, Y., Liu, Y., and Qiao, Y. Critic-guided decision transformer for offline reinforcement learning, 2023.

Wu, Y.-H., Wang, X., and Hamaya, M. Elastic decision transformer, 2023.

Zhuang, Z., Peng, D., Liu, J., Zhang, Z., and Wang, D. Reinformer: Max-return sequence modeling for offline rl, 2024.