Waypoint Transformer: Reinforcement Learning via Supervised Learning with Intermediate Targets

Anirudhan Badrinath **Yannis Flet-Berliac** **Allen Nie** **Emma Brunskill**
Department of Computer Science
Stanford University
`{abadrina, yfletberliac, anie, ebrun}@cs.stanford.edu`

Abstract

Despite the recent advancements in offline reinforcement learning via supervised learning (RvS) and the success of the decision transformer (DT), DTs have fallen short in several challenging benchmarks. The root cause of this underperformance lies in their inability to seamlessly connect segments of suboptimal trajectories. To overcome this limitation, we present a novel approach to enhance RvS methods by integrating intermediate goal targets. We introduce the Waypoint Transformer (WT), building upon the DT framework and conditioned on automatically-generated waypoints. The results show a significant increase in the final return compared to existing RvS methods, with performance on par or greater than existing state-of-the-art value-based methods.

1 Introduction

Traditionally, offline reinforcement learning (RL) methods that compete with state-of-the-art (SOTA) algorithms have relied on objectives encouraging pessimism in combination with value-based methods (1; 2). However, these can be challenging to train, requiring intricate hyperparameter tuning and tricks to ensure stability and optimal performance.

Reinforcement learning via supervised learning (RvS) has emerged as a simpler alternative to traditional methods (3). RvS methods are based on behavioral cloning (BC) to train a policy and eliminate the need for any temporal-difference (TD) learning (e.g., fitted value functions). This yields a simpler algorithmic framework based on supervised learning. There are several successful applications of RvS, including goal-conditioned methods (4; 5; 6; 7; 3).

However, RvS methods, such as the decision transformer (7), have typically struggled in tasks with clear goals, where seamlessly connecting (or "stitching") appropriate segments of suboptimal training trajectories is critical for success (8). For example, when tasked with reaching goal locations in AntMaze environments or completing a series of tasks in FrankaKitchen with only a singular global goal, RvS typically performs significantly worse than TD learning methods (9; 10; 11).

We introduce a waypoint generation technique that produces intermediate goals, which serve as guidance to steer a policy to desirable goal outcomes. By conditioning a transformer-based RvS method adapted from DT on these generated targets, we obtain a trained policy that learns to follow them, leading to improved performance and stability compared to prior offline RL methods.

2 Preliminaries

We assume that there exists an agent interacting with a Markov decision process (MDP) with states $s_t \in \mathcal{S}$ and actions $a_t \in \mathcal{A}$ with unknown transition dynamics $p(s_{t+1} \mid s_t, a_t)$, initial state distribution $p(s_0)$, and global goal ω. The agent chooses an action from a transformer policy

$a_t \sim \pi_\theta(a_t \mid s_{t-k..t}, \Phi_{t-k..t}, \omega)$, parameterized by θ and conditioned on the known history of states $s_{t-k..t}$ and an additional proposed goal conditioning variable $\Phi_{t-k..t}$. To train our RvS algorithm on an offline dataset $\mathcal{D}$, we optimize the output of an autoregressive transformer model based on past and current states and conditioning variable, using a negative log-likelihood loss and gradient descent.

3 Methodology

3.1 Illustrative Example

To motivate the benefits of using waypoints, consider an infinite-horizon, deterministic MDP with $H+1$ states and two possible actions at non-terminal states. A graphical representation of the MDP is shown in Figure 1. For this scenario, we consider the goal-conditioned setting where the target goal state during train and test time is $\omega = s^{(H)}$, and the episode terminates once we reach ω.

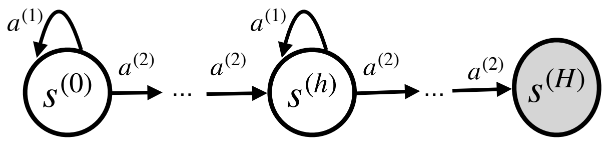

Figure 1: Chain MDP to motivate the benefit of intermediate goals for conditional BC-based policy training.

In offline RL, the data is often suboptimal for achieving the desired goal during testing. In this example, suppose we have access to a dataset $\mathcal{D}$ that contains an infinite number of trajectories collected by a random behavioral policy π_b where $\pi_b(a_t = a^{(1)} \mid s_t) = \lambda > 0$ for all s_t. Clearly, π_b is suboptimal with respect to reaching ω in the least number of timesteps; in expectation, it takes $\frac{H}{1-\lambda}$ timesteps to reach $s^{(H)}$ instead of H (optimal) since the agent "stalls" at the current state with probability λ and moves to the next state with probability $1-\lambda$.

Consider a global goal-conditioned policy $\pi_G(a_t \mid s_t, \omega)$ that is optimized using a behavioral cloning objective on $\mathcal{D}$. Clearly, the optimal policy $\pi_G^*(a_t \mid s_t, \omega) = \pi_b(a_t \mid s_t)\ \forall s_t$ since $\omega = s^{(H)}$ is a constant. Hence, the global goal-conditioned policy π_G^* is as suboptimal as the behavioral policy π_b.

Instead, suppose that we condition a policy $\pi_W(a_t \mid s_t, \Phi_t)$ on an intermediate goal state $\Phi_t = s_{t+K}$ for some chosen $K < \frac{1}{1-\lambda}$ (expected timesteps before π_b executes a_2), optimized using a behavioral cloning objective on $\mathcal{D}$. For simplicity, suppose our target intermediate goal state Φ_t for some current state $s_t = s^{(h)}$ is simply the next state $\Phi_t = s^{(h+1)}$. Based on data $\mathcal{D}$ from π_b, the probability of taking action $a^{(2)}$ conditioned on the chosen Φ_t and s_t is estimated as:

$$\begin{aligned}\Pr_{\pi_b}[a_t = a^{(2)} \mid s_t = s^{(h)}, s_{t+K} = s^{(h+1)}] &= \frac{\Pr_{\pi_b}[a_t = a^{(2)}, s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}]}{\Pr_{\pi_b}[s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}]} \\ &= \frac{(1-\lambda)\lambda^{K-1}}{\binom{K}{1}[(1-\lambda)\lambda^{K-1}]} = \frac{1}{\binom{K}{1}} = \frac{1}{K}.\end{aligned}$$

Hence, for the optimal intermediate goal-conditioned policy π_W^* trained on $\mathcal{D}$, the probability of choosing the optimal action $a^{(2)}$ is:

$$\pi_W^*(a_t = a^{(2)} \mid s_t = s^{(h)}, \Phi_t = s^{(h+1)}) = \frac{1}{K}.$$

Since $\pi_G^*(a_t = a^{(2)} \mid s_t = s^{(h)}, \omega) = 1-\lambda$ and we choose K such that $\frac{1}{K} > 1-\lambda$, we conclude:

$$\pi_W^*(a_t = a^{(2)} \mid s_t, \Phi_t) > \pi_G^*(a_t = a^{(2)} \mid s_t, \omega).$$

The complete derivation is presented in Appendix A. Based on this example, conditioning the actions on reaching a desirable intermediate state is more likely to result in taking the optimal action compared to a global goal-conditioned policy. Conditioning acts as a "guide" for the policy, directing it toward desirable intermediate goal targets in order to reach the global goal.

3.2 Augmenting Goal Conditioning with Waypoint Generation

In this section, we develop the waypoint network to address RvS's inability to "stitch" to achieve the desired goal, based on (8). Specifically, stitching requires considering experiences that are relevant to appropriate short-term goals. To illustrate this, we show the AntMaze Large environment in Figure 2, where the objective is to reach a target location from the start location (9). Analyzing the training trajectories that pass through either the start (blue) or target location (red), less than 5% of trajectories pass through both start and target regions; hence, the policy must "stitch" together subsequences from the blue and red trajectories within the stitching region. By providing intermediate targets within this region, rather than conditioning solely on the global goal, we can guide the policy to connect the relevant subsequences needed to reach the target.

To obtain effective intermediate targets, we propose the goal waypoint network, explicitly designed to generate short-term goals Φ_t. The purpose of these intermediate targets is to guide the policy network π_θ, conditioned on Φ_t, towards states that lead to the desired global goal by stitching relevant subsequences.

To that end, we represent the waypoint network W_ϕ, parameterized by ϕ, as a neural network that makes approximate K-step predictions of future observations Φ_t, conditioned on the current state, s_t, and target goal, ω. Formally, we minimize the objective in Equation 1 across the dataset $\mathcal{D}$, where L_ϕ is a mean-squared error for continuous state spaces:

$$\arg\min_{\phi} \sum_{\tau \in \mathcal{D}} L_\phi(W_\phi(s_t, \omega), s_{t+K}). \qquad (1)$$

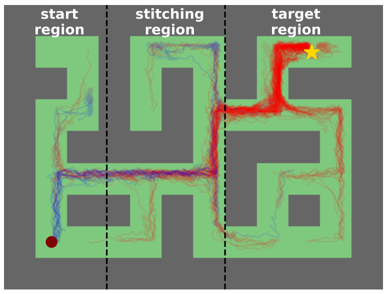

Figure 2: `antmaze-large` task to navigate from start (circle) to target (star). Blue and red colored lines are training trajectories passing through start or end locations respectively.

3.3 Waypoint Transformer

We propose the waypoint transformer (WT), a transformer-based offline RL method that leverages the proposed waypoint network W_ϕ and a GPT-2 architecture based on multi-head attention (12). The WT policy π_θ is conditioned on past states $s_{t-k..t}$ and goal waypoints $\Phi_{t-k..t} = W_\phi(s_{t-k..t}, \omega)$ with a context window of size k, as in Figure 3.

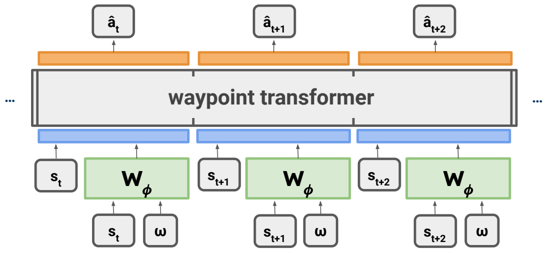

Figure 3: Waypoint Transformer architecture, where $\Phi_t = W_\phi(s_t, \omega)$ represents the output of the goal waypoint network.

We train the waypoint network W_ϕ on offline dataset $\mathcal{D}$ independently of the policy. To train the WT policy, we use gradient descent to iteratively optimize its parameters θ. To further simplify the design and improve computational efficiency, the WT is not conditioned on past actions $a_{t-k..t}$ (i.e., unlike the DT).

4 Experiments

We evaluate WT across challenging goal-conditioned tasks, with comparisons to prior offline RL methods. including conditional BC methods such as DT and RvS-R/G; value-based methods such as Onestep RL (13), TD3 + BC (14), CQL, and IQL; and standard BC baselines. For this, we leverage D4RL, an open-source benchmark for offline RL, consisting of varying datasets for tasks from AntMaze and FrankaKitchen (9). These environments present a challenge for offline RL methods as they contain no optimal trajectories and perform critical evaluations of a model's stitching ability (9). For instance, `partial` and `mixed` offline datasets in Kitchen consist of suboptimal, undirected data.

Across all tasks, Table 1 shows that WT (66.0 $\pm$ 4.4) improves upon the next best, IQL (59.8 $\pm$ 8.0), with respect to average score. In terms of variability across seeds, there is a notable reduction compared to IQL and most other methods. In the most challenging tasks requiring stitching, our method demonstrates performance far exceeding the next best method, IQL. On the AntMaze Large datasets, WT demonstrates a significant relative percentage improvement of 83.1% (`play`) and 51.6% (`diverse`). On Kitchen Partial and Mixed, the improvement is 37.8% and 39.0% respectively.

5 Utility of Waypoint Networks

To analyze the utility and behavior of waypoint networks, we qualitatively evaluate an agent's performance across rollouts of trained transformer policies on `antmaze-large-play-v2`. For this

Table 1: Normalized scores and training, where **blue** highlights indicates SOTA performance.

Environment	TD3 + BC	Onestep RL	CQL	IQL	BC	10% BC	RvS-R/G	DT	WT (Ours)
antmaze-umaze-v2	78.6	64.3	74.0	**87.5 ± 2.6**	54.6	62.8	65.4 ± 4.9	53.6 ± 7.3	64.9 ± 6.1
antmaze-umaze-diverse-v2	71.4	60.7	**84.0**	62.2 ± 13.8	45.6	50.2	60.9 ± 2.5	42.2 ± 5.4	71.5 ± 7.6
antmaze-medium-play-v2	10.6	0.3	61.2	**71.2 ± 7.3**	0.0	5.4	58.1 ± 12.7	0.0 ± 0.0	62.8 ± 5.8
antmaze-medium-diverse-v2	3.0	0.0	53.7	**70.0 ± 10.9**	0.0	9.8	**67.3 ± 8.0**	0.0 ± 0.0	**66.7 ± 3.9**
antmaze-large-play-v2	0.2	0.0	15.8	39.6 ± 5.8	0.0	0.0	32.4 ± 10.5	0.0 ± 0.0	**72.5 ± 2.8**
antmaze-large-diverse-v2	0.0	0.0	14.9	47.5 ± 9.5	0.0	6.0	36.9 ± 4.8	0.0 ± 0.0	**72.0 ± 3.4**
antmaze-avg-v2	27.3	20.9	50.6	63.0 ± 8.3	16.7	22.5	53.5 ± 7.2	16.0 ± 2.1	**68.4 ± 4.9**
kitchen-complete-v0	-	-	43.8	**62.5**	**65.0**	4.0	50.2 ± 3.6	46.5 ± 3.0	49.2 ± 4.6
kitchen-partial-v0	-	-	49.8	46.3	38.0	**66.0**	51.4 ± 2.6	31.4 ± 19.5	**63.8 ± 3.5**
kitchen-mixed-v0	-	-	51.0	51.0	51.5	40.0	60.3 ± 9.4	25.8 ± 5.0	**70.9 ± 2.1**
kitchen-avg-v0	-	-	48.2	53.3 ± 7.5	51.5	36.7	54.0 ± 5.2	34.6 ± 9.2	**61.3 ± 3.4**
average	-	-	49.8	59.8 ± 8.0	28.3	27.2	53.7 ± 6.5	22.2 ± 4.5	**66.0 ± 4.4**

analysis, we consider a WT policy (using a goal waypoint network with $K = 30$) and a global goal-conditioned transformer policy (i.e., no intermediate goals).

The ant's locations across 100 rollouts of a WT policy (Figure 4a) and a global goal-conditioned transformer policy (Figure 4b) demonstrate that WT shows notably higher ability and consistency in reaching the goal location. Without intermediate goals, the ant occasionally turns in the wrong direction and demonstrates a lesser ability to successfully complete a turn (Figure 4b). Consequently, the WT achieves more than twice the evaluation return (72.5 ± 2.8) compared to the global goal-conditioned policy (33.0 ± 10.3).

6 Discussion

In this study, we address the issues with existing conditioning techniques used in RvS, such as the "stitching" problem associated with global goals, through the automatic generation of intermediate targets. Based on empirical evaluations, we demonstrate significantly improved performance and stability compared to existing RvS methods, often on par with or outperforming TD learning methods. Especially on challenging tasks with suboptimal dataset composition, such as AntMaze Large and Kitchen Partial/Mixed, the guidance provided by the waypoint network through intermediate targets (e.g., as shown in Figure 4) significantly improves upon existing state-of-the-art performance. We believe that this work can present a pathway forward to developing practical offline RL methods leveraging the simplicity of RvS and exploring more effective conditioning techniques, as formalized by (3).

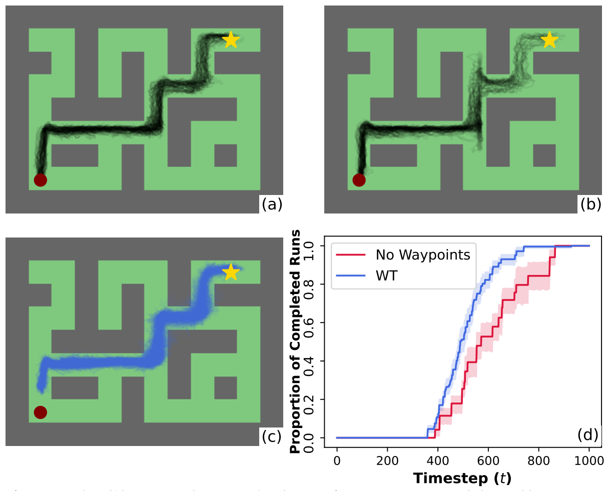

Figure 4: Shows the ant's location across 100 rollouts of **(a)** a WT policy and **(b)** a global goal-conditioned transformer policy; **(c)** generated intermediate goals by the waypoint network W_ϕ, **(d)** the proportion of all successful runs completed by timestep t.

However, despite improvements across challenging tasks, WT's margin of improvement on AntMaze U-Maze and Kitchen Complete (i.e., easier tasks) is lower. We believe this likely due to stitching being less necessary in such tasks compared to difficult tasks. Further characterizing the performance of WT on such tasks is an interesting direction for fuure work.

7 Conclusion

We propose a method for goal-conditioned reinforcement learning via supervised learning, Waypoint Transformer, conditioned on generated intermediate targets. We show that RvS with waypoints significantly surpasses existing RvS methods and achieves on par with or surpasses SOTA. We believe that WT advances the performance and applicability of RvS within the context of offline RL.

References

[1] 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.

[2] 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.

[3] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

[4] Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

[5] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[6] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[8] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? *arXiv preprint arXiv:2204.05618*, 2022.

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

[10] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[11] 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.

[12] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[13] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in neural information processing systems*, 34:4933–4946, 2021.

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

A Derivation for Illustrative Example

We provide detailed derivations based on the simple deterministic MDP shown in Section 4.1, in the context of an offline dataset $\mathcal{D}$ collected by a random behavioural policy π_b. We show that minimization of a maximum likelihood objective on π_G yields π_b, the behavioural policy. Note $\pi_G(a_t \mid s_t, \omega) = \pi_G(a_t \mid s_t)$ as $\omega = s^{(H)}$ is a constant (and as a result, a_t is conditionally independent). To obtain the optimal policy π_G^*, we maximize the following objective:

$$\arg\max_{\pi_G} \mathbb{E}_{(s_t,a_t)\in\mathcal{D}}[\log \pi_G(a_t \mid s_t, \omega)]$$

We simplify an expectation over an infinitely large dataset $\mathcal{D}$ collected by π_b:

$$\mathbb{E}_{(s_t,a_t)\in\mathcal{D}}[\log \pi_G(a_t \mid s_t, \omega)] = \mathbb{E}_{s_t\in\mathcal{D}}[\lambda \log \pi_G(a_t = a^{(1)} \mid s_t, \omega) + (1-\lambda)\log \pi_G(a_t = a^{(2)} \mid s_t, \omega)]$$

Since the actions are conditionally independent of the states, let $\hat{p} = \pi_G(a_t = a^{(1)} \mid s_t, \omega)$ for any state s_t. Then:

$$\mathbb{E}_{(s_t,a_t)\in\mathcal{D}}[\log \pi_G(a_t \mid s_t, \omega)] = \lambda \log \hat{p} + (1-\lambda)\log(1-\hat{p})$$

We can use calculus to maximize the above objective with respect to $\hat{p}$.

$$\begin{aligned}\frac{d}{d\hat{p}}\left[\lambda \log \hat{p} + (1-\lambda)\log(1-\hat{p})\right] &= \frac{\lambda}{\hat{p}} - \frac{1-\lambda}{1-\hat{p}} \\ &= \frac{\lambda(1-\hat{p}) - (1-\lambda)\hat{p}}{\hat{p}(1-\hat{p})}\end{aligned}$$

Setting the derivative to 0:

$$\lambda(1-\hat{p}) - (1-\lambda)\hat{p} = \lambda - \lambda\hat{p} - \hat{p} + \lambda\hat{p} = 0 \implies \boxed{\hat{p} = \lambda}$$

This yields an identical policy to the behavioural policy π_b. Next, consider the derivation of the probability of taking action $a^{(2)}$ conditioned on Φ_t and s_t based on data $\mathcal{D}$ from π_b:

$$\begin{aligned}&\Pr_{\pi_b}[a_t = a^{(2)} \mid s_t = s^{(h)}, s_{t+K} = s^{(h+1)}] \\ &= \frac{\Pr_{\pi_b}[a_t = a^{(2)}, s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}]}{\Pr_{\pi_b}[s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}]}\end{aligned}$$

In the above step, we used the definition of conditional probability. To compute these probabilities, we recognize that to end up with $s_{t+K} = s^{(h+1)}$ from $s_t = s^{(h)}$, the agent must take action $a^{(2)}$ exactly once between timestep t and $t+K-1$; any more implies the agent has moved beyond $s^{(h+1)}$ and any less implies the agent is still at $s^{(h)}$.

The probability in the numerator can be written as a product of taking action $a^{(2)}$ at timestep t, followed by taking action $a^{(1)}$ at timestep $t+1$ to $t+K-1$:

$$\begin{aligned}\Pr_{\pi_b}[a_t = a^{(2)}, s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}] &= (1-\lambda)\prod_{t'=t+1}^{t+K-1} \lambda \\ &= (1-\lambda)\lambda^{K-1}\end{aligned}$$

The probability in the denominator can be written as a product of taking action $a^{(2)}$ at exactly one timestep $t \le t' < t+K$, followed by taking action $a^{(1)}$ at the remaining timesteps. This can be modeled by a binomial probability where there are K slots to take action $a^{(1)}$, each with probability $1-\lambda$. Hence:

$$\begin{aligned}\Pr_{\pi_b}[s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)} &= \binom{K}{1}(1-\lambda)\lambda^{K-1} \\ &= K(1-\lambda)\lambda^{K-1}\end{aligned}$$

The overall probability is computed as:

$$\frac{\Pr_{\pi_b}[a_t = a^{(2)}, s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}]}{\Pr_{\pi_b}[s_{t+K} = s^{(h+1)} \mid s_t = s^{(h)}]} = \frac{(1-\lambda)\lambda^{K-1}}{K(1-\lambda)\lambda^{K-1}} = \frac{1}{K}$$

We can apply a similar argument to show that π^*_W (i.e., at optimum) must clone the derived probability when a maximum likelihood objective is applied. Hence, for the optimal intermediate goal-conditioned policy π^*_W, we know it obeys:

$$\pi^*_W(a_t = a^{(2)} \mid s_t = s^{(h)}, \Phi_t = s^{(h+1)}) = \frac{1}{K}$$

Since $\pi^*_G(a_t = a^{(2)} \mid s_t = s^{(h)}, \omega) = \pi_b(a_t = a^{(2)} \mid s_t = s^{(h)}) = 1 - \lambda$ and we choose $K < \frac{1}{1-\lambda} \implies \frac{1}{K} > 1 - \lambda$, we conclude that:

$$\pi^*_W(a_t = a^{(2)} \mid s_t, \Phi_t) > \pi^*_G(a_t = a^{(2)} \mid s_t, \omega)$$

This concludes the derivation.

B Experimental Details

In this section, we provide more details about the experiments, including hyperparameter configuration, sources of reported results for each method, and details of each environment (i.e., version). For all experiments on WT, the proposed method, we run 5 trials with different random seeds and report the mean and standard deviation across them. On AntMaze and Kitchen, we use goal-conditioning. For all experiments on DT, we run 5 trials with random initializations using the default hyperparameters proposed in (7) and used in the official GitHub repository. We are unable to reproduce some of the results demonstrated in (7) and reported in succeeding work such as (10; 3).

B.1 Environments and Tasks

AntMaze For AntMaze tasks, we include previously reported results for all methods except RvS-G from (10). The results for the RvS-G are from (3). We run experiments for DT (reward-conditioned, as per (7)) and WT across 5 seeds. For all reported results, including WT, AntMaze v2 is used as opposed to AntMaze v0.

FrankaKitchen On Kitchen, we include available reported results from (10) for all methods except RvS-G and (3) for RvS-G, with results omitted for TD3 + BC and Onestep RL as they are not available in other work or provided by the authors. Similarly to AntMaze, we run experiments for DT and WT across 5 seeds. The target goal configuration for WT is "all" (i.e., where all the tasks are solved), per (3). For all reported results, including WT, Kitchen v0 is used.

B.2 WT Hyperparameters

In Table 2, we show the chosen hyperparameter configuration for WT across all experiments. Consistent with the neural network model in RvS-R/G with 1.1M parameters (3), the WT contains 1.1M trainable parameters. For the most part, the chosen hyperparameters align closely with default values in deep learning; for example, we use the ReLU activation function and a learning rate of 0.001 with the Adam optimizer.

In Table 3, we show the chosen hyperparameter configuration for the goal waypoint networks across all experiments. In general, the goal waypoint network outputs the same dimension as the state since it makes k-step predictions. Depending on the environment, the goal waypoint outputs either a 2-dimensional location for AntMaze or a 30-dimensional state for Kitchen.

C Ablation Studies

Waypoint Network On goal-conditioned tasks, we examine the behavior of the goal waypoint network as it relates to the performance of the policy at test time by ablating aspects of its configuration

Table 2: Hyperparameters and configuration details for WT across all experiments.

Hyperparameter	**Value**
Transformer Layers	2
Transformer Heads	16
Dropout Probability (attn)	0.15
Dropout Probability (resid)	0.15
Dropout Probability (embd)	0.0
Non-Linearity	ReLU
Learning Rate	0.001
Gradient Steps	30,000
Batch Size	1024

Table 3: Hyperparameters and configuration details for goal waypoint networks across all experiments.

Hyperparameter	**Value**
Number of Layers	3
Dropout Probability	0.0
Non-Linearity	ReLU
Learning Rate	0.001
Gradient Steps	40,000
Batch Size	1024

and training. For this analysis, we consider `antmaze-large-play-v2`, a challenging task that critically evaluates the stitching capability of offline RL techniques.

To understand the effect of the configuration of the goal waypoint network on test performance, we ablate two variables relevant to generating effective intermediate goals: the temporal proximity of intermediate goals (K) and the validation loss of the goal waypoint network. Additionally, we perform comparisons between the goal waypoint network and manually constructed waypoints, for which the methodology and results are shown in Appendix D.

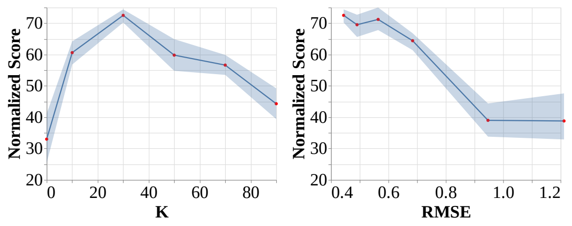

Figure 5: Normalized score attained by WT on `antmaze-large-play-v2` based on varying **left**: the temporal proximity of generated goals, K, and **right**: goal waypoint network RMSE on a held-out dataset.

The normalized score attained by the agent is shown as a function of K and the validation loss of the goal waypoint network in Figure 5. For this environment and dataset, an ideal choice for K is around 30 timesteps. For all nonzero K, the performance is reduced at a reasonably consistent rate on either side of $K = 30$. Importantly, when $K = 0$ (i.e., no intermediate goals), there is a notable reduction in performance compared to all other choices of K; compared to the optimal $K = 30$, the score is reduced by a factor of 2.2x.

In Figure 5 (right), the normalized score shows the negligible change for values of held-out RMSE between 0.4 and 0.6, corresponding to at least 1,000 gradient steps or roughly 30 sec of training, with a sharper decrease henceforth. As the RMSE increases to over 1, we observe a relative plateau in performance near an average normalized score of 35-45, roughly corresponding to performance without using a waypoint network (i.e., $K = 0$ in Figure 5 (left)).

Transformer Configuration Based on the work in (3), we balance between expressiveness and regularization to maximize policy performance. We ablate the probability of node dropout p_{drop} and the number of transformer layers L. To further examine this balance, we experiment with

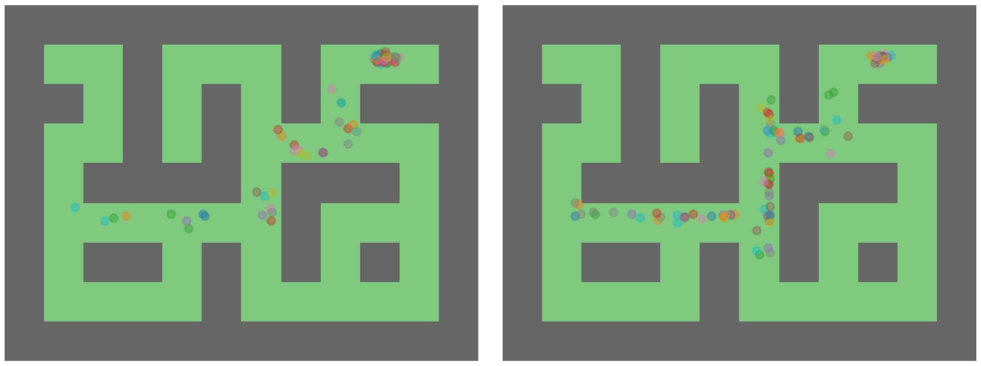

Figure 6: End locations for `antmaze-large-play-v2` during 100 rollouts of **left**: WT and **right**: global goal-conditioned transformer policy.

conditioning on past actions $a_{t-k..t-1}$, similarly to the DT, to characterize its impact on performance and computational efficiency. Similarly to previous sections, we consider `antmaze-large-play-v2`.

Based on Table 4, we observe that the sensitivity to the various ablated hyperparameters is relatively low in terms of performance, and removing action conditioning results in reduced training time and increased performance. In the context of prior RvS work where dropout ($p_{\text{drop}} = 0.1$) decreased performance compared to no dropout ($p_{\text{drop}} = 0.0$) by 1.5-3x on AntMaze, the largest decrease in average performance on WT is only by a factor of 1.1x (3).

Table 4: Ablation study of transformer configuration with normalized score on `antmaze-large-play-v2`, including dropout probability (p_{drop}), number of transformer layers (L), and conditioning on actions, where bolded selections are used for final models.

p_{drop}	**Score**
0.000	68.3 ± 5.9
0.075	70.8 ± 4.5
0.150	72.5 ± 2.8

Conditioning	**Score**	**Runtime**
$(s_{t-k..t}, a_{t-k..t-1})$	66.5 ± 5.6	30 min
$\mathbf{s_{t-k..t}}$ **(no actions)**	72.5 ± 2.8	20 min

L	**Score**
1	72.1 ± 5.7
2	72.5 ± 2.8
3	71.8 ± 3.0

D Additional Experiments

D.1 Analysis of Stitching Region Behavior

To add on to the analysis of the goal waypoint network presented in the main text, we analyze the "failure" regions of transformer policies with and without a goal waypoint network. That is, by determining the final locations of the agent, we can examine where the agent ended up instead of the target location. Similar to the analysis in Section 6.3, this analysis can inform the stitching capability of our methods.

Based on Figure 6, it is clear that the WT does not get "stuck" (e.g., after taking the wrong turn) as often as the policy conditioned on global-goals. Moreover, the number of ants ending up near the beginning portions of the maze (i.e., the bottom left) is significantly smaller for WT, which contributes to its doubled success rate. We believe these are primarily attributable to the guidance provided by the goal waypoint network through a consistent set of intermediate goals to reach the target location at evaluation time.

Interestingly, we observe that WT displays an increased rate of failure around the final turn relative to other regions in the maze. As there is a relative lack of density in other failure regions closer to the beginning of the maze, we hypothesize that some rollouts may suffer from the ant getting "stuck" at challenging critical points in the maze, as defined in (8). This indicates an interesting direction of exploration for future work and a technique to combat this could result in policies with nearly 100% success rate in completing `antmaze-large-play-v2`.

D.2 Comparisons to Manual Waypoint Selection

We compare the performance of the proposed goal waypoint network with a finite set of manual waypoints, hand-selected based on prior oracular knowledge about the critical points within the maze for achieving success (i.e., turns, midpoints). Based on the selected manual waypoints, shown in Figure 7, we use a simple algorithm to provide intermediate targets Φ_t based on a distance-based sorting approach, shown in Algorithm 1.

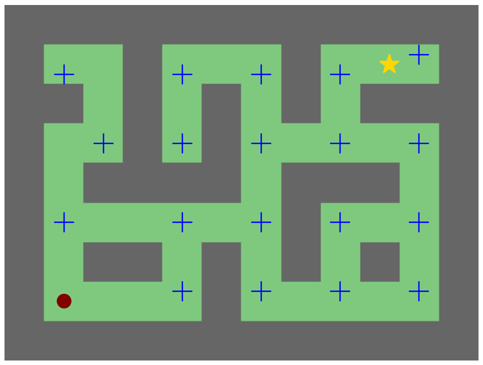

Figure 7: Manually selected waypoints (blue pluses) for `antmaze-large-play-v2`, the chosen task to evaluate the proposed approach. As before, the start location is marked with a maroon dot, and the target location is marked wit a gold star.

Algorithm 1 Manual waypoint selection with W_m and s_t using L_2 distance and a given global goal ω.

```
$W_c \leftarrow \{w_m : ||w_m - \omega||_2 \leq ||s_t - \omega||_2\}$ {consider waypoints that brings agent closer to $\omega$}
return $\arg\min_{w_c \in W_c} ||w_c - s_t||_2$
```

With all configuration and hyperparameters identical to WT, we compare the performance of a global goal-conditioned policy, WT with manual waypoints, and WT with the goal waypoint network on `antmaze-large-play-v2` in Table 5.

The results demonstrate that WT clearly outperforms manual waypoint selection in succeeding in the AntMaze Large environment. However, while comparing a global-goal conditioned policy and a policy conditioned on manual waypoints, it is clear that the latter improves upon average performance and variability across initialization seeds. We believe that this illustrates that (a) waypoints, whether manual or generated, tend to improve performance of the policy and (b) finer-grained waypoints provide more valuable information for the policy to succeed more.

Table 5: Normalized evaluation scores for different waypoint selection techniques on the `antmaze-large-play-v2` task.

Technique	**Normalized Score**
No Waypoints	33.0 ± 10.3
Manual Waypoints	44.5 ± 2.8
Waypoint Network	$\mathbf{72.5 \pm 2.8}$

We believe that this provides further verification and justification for both the generation of intermediate targets and the procedure of generation through a goal waypoint network that performs k-step prediction.