Waypoint Transformer: Reinforcement Learning via Supervised Learning with Intermediate Targets (Appendix)

Anonymous Author(s)
Affiliation
Address
email

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 that $\pi_G(a_t \mid s_t, \omega) = \pi_G(a_t \mid s_t)$ 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:

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

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

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 $t \leq t' < t + K$, 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:

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

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, whereas reward-conditioning is used for Gym-MuJoCo. For all experiments on DT, including Gym-MuJoCo, we run 5 trials with random initializations using the default hyperparameters proposed in Chen et al. [2021] and used in the official GitHub repository[1]. We are unable to reproduce some of the results demonstrated in Chen et al. [2021] and reported in succeeding work such as Kostrikov et al. [2021], Emmons et al. [2021].

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

B.1 Environments and Tasks

AntMaze. For AntMaze tasks, we include previously reported results for all methods except RvS-G from Kostrikov et al. [2021]. The results for the RvS-G are from Emmons et al. [2021]. We run experiments for DT (reward-conditioned, as per Chen et al. [2021]) 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 Kostrikov et al. [2021] for all methods except RvS-G and Emmons et al. [2021] 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 Emmons et al. [2021]. For all reported results, including WT, Kitchen v0 is used.

Gym-MuJoCo. On the evaluated locomotion tasks, we use reported results from Kostrikov et al. [2021] for all methods except RvS-R and Emmons et al. [2021] (RvS-R). We run experiments for DT and WT across 5 seeds. The MuJoCo v2 environments are used for all methods.

B.2 WT Hyperparameters

In Table 1, we show the chosen hyperparameter configuration for WT across all experiments, run on a GTX 3090 GPU, an 8-core CPU, and 16 GB of RAM. Consistent with the neural network model in RvS-R/G with 1.1M parameters [Emmons et al., 2021], 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.

Table 1: 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

In Table 2, we show the chosen hyperparameter configuration for the reward and goal waypoint networks across all experiments. The reward waypoint network always outputs 2 values, the ARTG and CRTG. 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.

Table 2: Hyperparameters and configuration details for goal and reward 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

B.3 Evaluation Return Targets

The target returns for the Gym-MuJoCo tasks are specified in Table 3, in the form of normalized scores. These were obtained typically by performing exhaustive grid searches over 4-6 candidate target return values, following prior work [Chen et al., 2021, Emmons et al., 2021]. Typically, we choose the range of the grid search based on the interval close to or higher than the state-of-the-art normalized scores on each of the tasks.

Table 3: Normalized score targets for WT on reward-conditioned tasks in Gym-Mujoco.

Task	Normalized Score Target
hopper-medium-replay-v2	95
hopper-medium-v2	73.3
hopper-medium-expert-v2	125
walker2d-medium-replay-v2	90
walker2d-medium-v2	85
walker2d-medium-expert-v2	122.5
halfcheetah-medium-replay-v2	45
halfcheetah-medium-v2	52.5
halfcheetah-medium-expert-v2	105

C Analysis of Bias and Variance of Reward-Conditioning Variables

We analyze the bias and variance of existing reward-conditioning techniques: a constant average reward-to-go (ARTG) target, as used in Emmons et al. [2021], and a cumulative return target updated with rewards collected during the episode (CRTG), as in Chen et al. [2021]. By analyzing the bias and variance of these techniques, we can determine the potential issues that may explain the performance of methods that condition using these techniques.

Consider the definitions of the true ARTG (R_a) and CRTG (R_c) below, based on a given trajectory τ where the length of the trajectory $|\tau| = T$. These definitions are used to train the policy on offline data $\tau \in \mathcal{D}$.

$$R_a(\tau, t) = \frac{1}{T-t} \sum_{t'=t}^{T} \gamma^{t'} r_{t'} \tag{1}$$

$$R_c(\tau, t) = \sum_{t'=t}^{T} \gamma^{t'} r_{t'} \tag{2}$$

At evaluation time, it is impossible to calculate $r_{t'}$ for any $t' \geq t$. As a result, we provide ARTG and CRTG targets, θ_a and θ_c respectively. At evaluation time, the values of the ARTG and CRTG are estimated and used as follows for each timestep t in Chen et al. [2021] and Emmons et al. [2021]:

$$\hat{R}_a(\tau, t) = \theta_a \tag{3}$$

$$\hat{R}_c(\tau, t) = \theta_c - \sum_{t'=1}^{t} \gamma^{t'} r_{t'} \tag{4}$$

Ideally, the errors of the estimated $\hat{R}_a$ and $\hat{R}_c$ are minimal so as to accurately approximate the true average or cumulative reward-to-go respectively, but that is often infeasible. To characterize the error of each of the evaluation estimates of ARTG and CRTG, consider the decomposition of the error for a particular reward conditioning variable R and estimated evaluation $\hat{R}$ presented in Theorem C.1.

Theorem C.1. *The general bias-variance decomposition of the expected squared error between a true $R(\tau, t)$ (e.g., Equations 1 or 2) and an estimator $\hat{R}(\tau, t)$ (e.g., Equations 3 or 4) under the*

distribution of trajectories τ induced by an arbitrary policy π and unknown transition dynamics $p(s_{t+1} \mid s_t, a_t)$ is given by:

$$\mathbb{E}_\tau[(R(\tau,t) - \hat{R}(\tau,t))^2] = \mathbb{E}[R(\tau,t) - \hat{R}(\tau,t)]^2 + \mathrm{Var}[\hat{R}(\tau,t) - R(\tau,t)].$$

Proof. Similarly to the derivation of the standard bias-variance tradeoff, we expand terms and separate them into multiple expectations using the linearity of expectation. We leverage the definition of the covariance, $\mathrm{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$, and variance, $\mathrm{Var}[X] = E[X^2] - E[X]^2$, in several steps.

$$\mathbb{E}_\tau[(R(\tau,t) - \hat{R}(\tau,t))^2] = \mathbb{E}[R(\tau,t)^2] + \mathbb{E}[\hat{R}(\tau,t)^2] - 2\mathbb{E}[\hat{R}(\tau,t)R(\tau,t)].$$

We can simplify the first term using the definition of the variance and the third term using the definition of the covariance.

$$\begin{aligned}\mathbb{E}_\tau[(R(\tau,t) - \hat{R}(\tau,t))^2] &= \mathrm{Var}[R(\tau,t)] + \mathbb{E}[R(\tau,t)]^2 + \mathbb{E}[\hat{R}(\tau,t)^2] - 2\mathbb{E}[\hat{R}(\tau,t)R(\tau,t)] \\ &= \mathrm{Var}[R(\tau,t)] + \mathbb{E}[R(\tau,t)]^2 + \mathbb{E}[\hat{R}(\tau,t)^2] - 2(\mathrm{Cov}(\hat{R}(\tau,t), R(\tau,t)) + \\ &\quad \mathbb{E}[R(\tau,t)] \cdot \mathbb{E}[\hat{R}(\tau,t)])).\end{aligned}$$

Similarly, we simplify the $\mathbb{E}[\hat{R}(\tau,t)^2]$ term using the definition of the variance and collect terms.

$$\begin{aligned}\mathbb{E}_\tau[(R(\tau,t) - \hat{R}(\tau,t))^2] &= (\mathbb{E}[R(\tau,t)] - \mathbb{E}[\hat{R}(\tau,t)])^2 + \mathrm{Var}[\hat{R}(\tau,t)] - 2\mathrm{Cov}(\hat{R}(\tau,t), R(\tau,t)) + \\ &\quad \mathrm{Var}[R(\tau,t)] \\ &= \mathbb{E}[R(\tau,t) - \hat{R}(\tau,t)]^2 + \mathrm{Var}[\hat{R}(\tau,t)] - 2\mathrm{Cov}(\hat{R}(\tau,t), R(\tau,t)) + \\ &\quad \mathrm{Var}[R(\tau,t)].\end{aligned}$$

Equivalently, since $\mathrm{Var}[X - Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] - 2\mathrm{Cov}(X,Y)$, we can rewrite the result as follows.

$$\mathbb{E}_\tau[(R(\tau,t) - \hat{R}(\tau,t))^2] = \mathbb{E}[R(\tau,t) - \hat{R}(\tau,t)]^2 + \mathrm{Var}[\hat{R}(\tau,t) - R(\tau,t)].$$

This completes the derivation of the general bias-variance decomposition. ■

Analysis of ARTG. Consider the decomposition of the error per the bias-variance tradeoff of the ARTG. Trivially, the variance of the estimate in Equation 3 is zero and it is independent of R because it is a constant value, and as a result, the expected error is composed entirely of the bias and irreducible variance of the true ARTG.

$$\begin{aligned}E_\tau[(R_a(\tau,t) - \hat{R}_a(\tau,t))^2] &= E_\tau[R_a(\tau,t) - \hat{R}_a(\tau,t)]^2 + \mathrm{Var}[R_a(\tau,t)] \\ &= E_\tau[R_a(\tau,t) - \theta_a]^2 + \mathrm{Var}[R_a(\tau,t)]\end{aligned}$$

Based on the terms that we are able to minimize, we can derive through calculus that the squared error and bias of a constant estimator are minimized by the mean, i.e., when $\hat{\theta}_a = \mathbb{E}_\tau[R_a(\tau,t)]$. However, in the offline RL setting, we cannot easily determine this value for an arbitrary trained policy π. As a result, the bias of the technique during evaluation can lead to high error in estimating the true ARTG, which may consequently cause reduced performance during the evaluation of the policy.

Empirically, we demonstrate that the high bias of this technique may lead to instability in achieved return across rollouts on `hopper-medium-replay-v2`. Specifically, suppose that a rollout is classified as unsuccessful if it terminates before the time limit $T = 1000$, and analogously, a successful rollout reaches $t = T$ without termination. Based on these distinctions, we display the true ARTG across 200 successful and unsuccessful rollouts of a trained transformer policy in Figure 1.

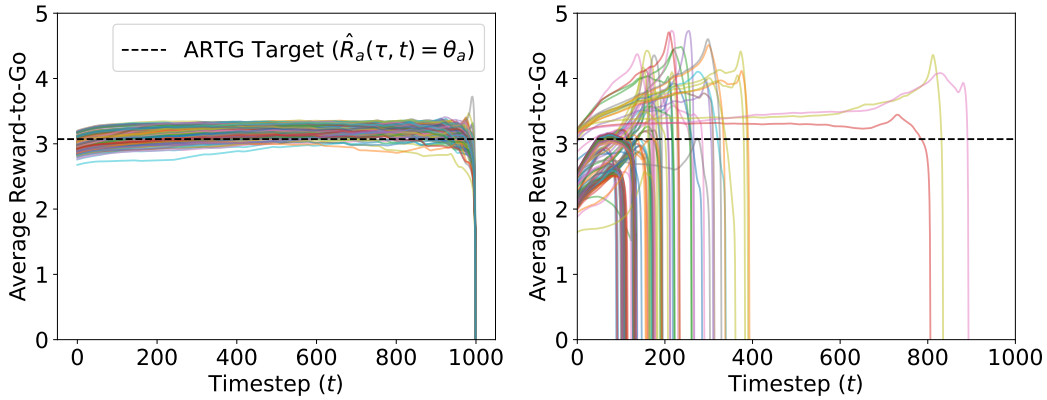

Figure 1: True average reward-to-go as a function of timestep t for 200 rollouts of a transformer policy on `hopper-medium-replay-v2` compared to the constant ARTG target θ_a (dotted black line), for **left**: successful rollouts and **right**: unsuccessful rollouts.

Clearly, for all successful rollouts, the true ARTG matches the constant target closely across most $t \in [1, T)$, with the exception of $t \approx T$. However, across most failed rollouts, the bias of the constant target for small $t \approx 0$ (i.e., when the policy is taking its first few actions) is relatively high. As t grows, the bias seems to spike upwards significantly and the episode terminates shortly thereafter, indicating an unsuccessful rollout.

Hence, it is evident that when the true ARTG closely follows the prescribed constant ARTG target, the policy consistently achieves state-of-the-art performance. However, whenever the target ARTG underestimates or overestimates the true ARTG by a margin of greater than 0.4, the rollout tends to be unsuccessful, often achieving less than half the return compared to successful rollouts.

To that end, the reward waypoint network uses a neural network formulation to estimate R_a (i.e., without using a constant estimator θ_a) based on the state s_t and the target return ω. By training the neural network to provide a less biased estimate of the ARTG, we show that we can achieve substantial performance improvements over a constant estimate of ARTG on the same task. As shown in Table 4, RvS-R and a constant ARTG both exhibit significantly lower average performance and greater variability compared to WT (with a reward waypoint network).

Table 4: Normalized evaluation scores for different policies and ARTG estimation techniques on the `hopper-medium-replay-v2` task.

Technique	Normalized Score
Transformer (constant ARTG)	66.5 ± 15.6
RvS-R (constant ARTG)	73.5 ± 12.8
WT (waypoint network)	$\mathbf{88.9 \pm 2.4}$

Analysis of CRTG. Consider a similar decomposition of the error using Theorem C.1 of the CRTG. Though neither the bias nor the variance of $\hat{R}_c$ are necessarily zero, it is important to note that the variance term can be large even if the bias is zero, i.e., if $\theta_c = \mathbb{E}[\sum_{t=1}^{T} \gamma^t r_t]$.

Corresponding to a best-case scenario where $\hat{R}_c$ is unbiased, we consider a bound of the expected error constructed using only the variance term. For simplicity, we assume that we incur the worst-case variance term with minimal covariance between R_c and $\hat{R}_c$. The resulting dominant quantities are the variance of $\hat{R}_c$ and the irreducible variance of R_c.

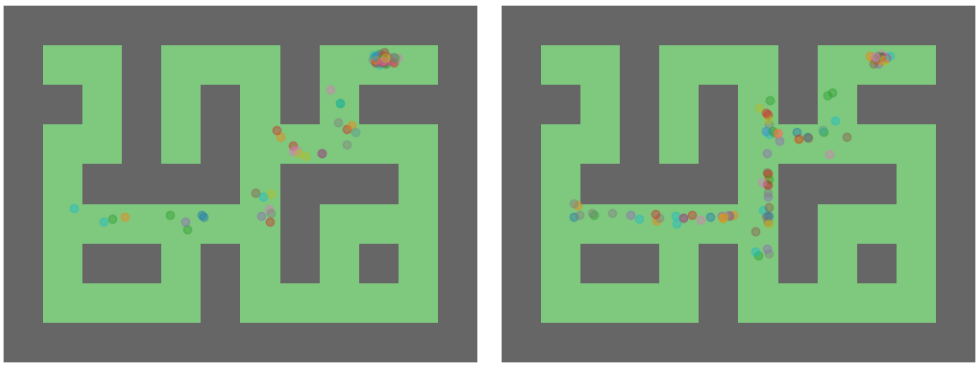

Figure 3: End locations for `antmaze-large-play-v2` during 100 rollouts of **left**: WT and **right**: global goal-conditioned transformer policy.

$$
\begin{aligned}
E_\tau[(R_c(\tau,t) - \hat{R}(\tau,t))^2] &= \mathbb{E}[R(\tau,t) - \hat{R}(\tau,t)]^2 + \mathrm{Var}[\hat{R}(\tau,t) - R(\tau,t)] \\
&\geq \mathrm{Var}[\hat{R}(\tau,t) - R(\tau,t)] \\
&\approx \mathrm{Var}[\sum_{t'=1}^{t} \gamma^{t'} r_{t'}] + \mathrm{Var}[R(\tau,t)]
\end{aligned}
$$

Prior work has shown that Monte Carlo estimates of return exhibit high variance, motivating techniques such as using baseline networks for REINFORCE, k-step returns or TD(λ) for TD-learning methods, etc., which aim to reduce variance at the cost of potentially incurring bias [Sutton and Barto, 1999]. In that vein, we demonstrate that our reward waypoint network predicts a similar quantity as a baseline network and inherits many of its desirable properties.

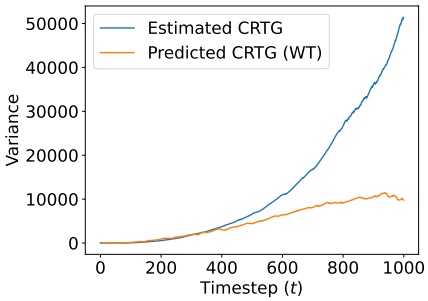

Figure 2: Variance of estimated CRTG (Equation 4) and predicted CRTG by the reward waypoint network on the `hopper-medium-replay-v2`.

Specifically, while the baseline network predicts $\sum_{t'=t}^{T} \gamma^t r_t$ given s_t, we additionally condition our reward waypoint network's prediction on ω. In the offline RL setting where datasets are often a mixture of suboptimal and optimal trajectories, we require that the waypoint network can differentiate between such trajectories. Hence, by conditioning on the overall reward ω, the reward waypoint network is able to differentiate between high and low return-achieving trajectories.

Empirically, we demonstrate that the predicted CRTG from our reward waypoint network exhibits lesser variance on `hopper-medium-replay-v2` than the estimated CRTG (i.e., as in Equation 4). As shown in Figure 2, the variance of both techniques are relatively similar until $t \approx 400$, after which the variance of the estimated CRTG appears to grow superlinearly as a function of t. In the worst case, at $t \approx T$, the variance is nearly 5x larger than for the predicted CRTG.

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 3, it is clear that the WT does not get "stuck" (e.g., after taking the wrong turn) as often as the policy that is 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 Kumar et al. [2022]. 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 Target Reward Interpolation

Unlike traditional value-based approaches, RvS methods such as DT, RvS, and WT present a simple methodology to condition the policy to achieve a particular target return. In theory, this allows RvS to achieve performance corresponding to the desired performance level, provided that it has non-zero coverage within the offline training dataset.

To examine RvS's capability in this regard, we replicate an analysis performed in Emmons et al. [2021] to examine whether RvS can interpolate between modes in the data to achieve a particular target return. In this case, we compare WT to RvS-R on the `walker2d-medium-expert-v2` task, in which the data is composed of two modes of policy performance (i.e., medium and expert). The mode corresponding to the "medium" data is centered at a normalized score of 80, whereas "expert" performance is located at a normalized score of 110.

Based on Figure 4, WT shows a reasonably improved ability to interpolate in some regions than RvS-R. Where RvS-R displays a failure to interpolate from normalized targets of 30-60 and WT does not, both tend to be unable to interpolate between 70-100 (i.e., between the modes of the dataset).

Although the reasons for the failure in interpolation are unclear across two RvS methods, it is a worthwhile analysis for future work. We hypothesize that techniques such as oversampling may mitigate this issue as this may simply be linked to low frequency in the data distribution.

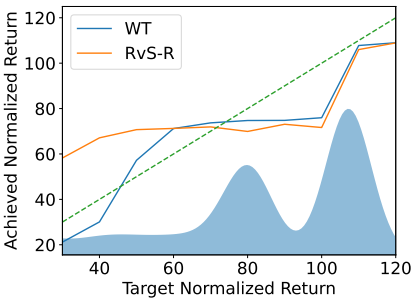

Figure 4: Achieved normalized score versus the target normalized score for `walker2d-medium-expert-v2` task on WT and RvS-R (green line is $y = x$).

D.3 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 5, we use a simple algorithm to provide intermediate targets Φ_t based on a distance-based sorting approach, shown in Algorithm 1.

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

procedure SELECT_WAYPOINTS(W_m, s_t, ω)
  $W_c \leftarrow \{w_m : ||w_m - \omega||_2 \leq ||s_t - \omega||_2\}$ ▷ consider waypoints that brings agent closer to ω
  return $\arg\min_{w_c \in W_c} ||w_c - s_t||_2$
end procedure

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

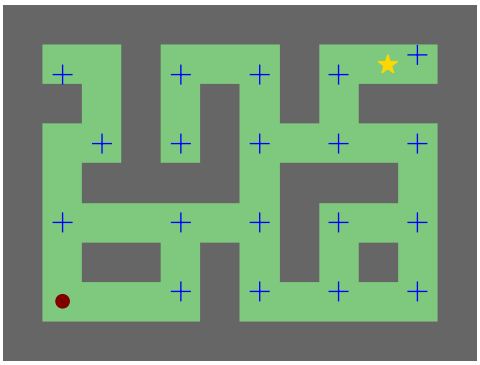

Figure 5: Manually selected waypoints (blue pluses) for `antmaze-large-play-v2`, the chosen task to evaluate the proposed approach. As before, the start location is marked with a maroon dot, and the target location is marked wit a gold star.

performance and variability across initialization seeds. These results suggest two important points: (a) whether manual or generated, waypoints generally improve policy performance, 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	$\mathbf{72.5} \pm \mathbf{2.8}$

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.

References

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.

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.

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

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.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. *Robotica*, 17(2): 229–235, 1999.