A Analysis of the conservatism term

With the goal of understanding the behavior of our training procedure, we theoretically analyze the solution obtained by Eq. 2 for the simpler cases when Q is represented as a table, and when the objective in Eq. 2 can be minimized exactly. We derive the minimizer of the objective in Eq. 2 by differentiating J with respect to Q:

$$\begin{aligned}
&\forall s, a, k, \quad \frac{dJ}{dQ(s,a)} = 0 \\
&\pi_\beta(a|s)\left(Q(s,a) - \mathcal{B}^* Q^k(s,a)\right) + \alpha\tilde{\pi}_\beta(a|s)Q(s,a) = 0 \\
&Q(s,a)\left(\pi_\beta(a|s) + \alpha\tilde{\pi}_\beta(a|s)\right) = \pi_\beta(a|s)\mathcal{B}^* Q^k(s,a) \\
&Q^{k+1}(s,a) = \underbrace{\frac{\pi_\beta(a|s)}{\pi_\beta(a|s) + \alpha\tilde{\pi}_\beta(a|s)}}_{:=m(s,a)} \cdot \mathcal{B}^* Q^k(s,a)
\end{aligned} \tag{3}$$

Eq. 3 implies that training with the objective in Eq. 2 performs weighted Bellman backup: unlike the standard Bellman backup, training with Eq. 2 multiplies large Q-value targets by a weight $m(s,a)$. This weight $m(s,a)$ takes values between 0 and 1, with larger values close to 1 for in-distribution actions where $(s,a) \in \mathcal{D}$, and very small values close to 0 for out-of-distribution actions a at any state s (i.e., actions where $\pi_\beta(a|s)$ is small). Thus, the Bellman backup induced via Eq. 3 should effectively prevent over-estimation of Q-values for unseen actions.

B Q-Transformer Architecture & System

In this section, we describe the architecture of Q-Transformer as well as the important implementation and system details that make it an effective Q-learning algorithm for real robots.

B.1 Transformer sequence model architecture

Our neural network architecture is shown in Figure 3. The architecture is derived from RT-1 design [1], adapted to accommodate the Q-Transformer framework, and consists of a Transformer backbone that reads in images via a convolutional encoder followed by tokenization. Since we apply Q-Transformer to a multi-task robotic manipulation problem where each task is specified by a natural language instruction, we first embed the natural language instruction into an embedding vector via the Universal Sentence Encoder [66]. The embedding vector and images from the robot camera are then converted into a sequence of input tokens via a FiLM EfficientNet [67, 68]. In the standard RT-1 architecture [1], the robot action space is discretized and the Transformer sequence model outputs the logits for the discrete action bins per dimension and per time step. In this work, we extend the network architecture to use Q-learning by applying a sigmoid activation to the output values for each action, and interpreting the resulting output after the sigmoid as Q-values. This representation is particularly suitable for tasks with sparse per-episode rewards $R \in [0, 1]$, since the Q-values may be interpreted as probabilities of task success and should always lie in the range $[0, 1]$. Note that unlike the standard softmax, this interpretation of Q-values does *not* prescribe normalizing *across* actions (i.e., *each* action output can take on any value in $[0, 1]$).

Since our robotic system, described in Section B.3, has 8-dimensional actions, we end up with 8 dimensions per time step and discretize each one into $N = 256$ value bins. Our reward function is a sparse reward that assigns value 1.0 at the last step of an episode if the episode is successful and 0.0 otherwise. We use a discount rate $\gamma = 0.98$. As is common in deep RL, we use a target network to estimate target Q-values Q^k, using an exponential moving average of model weights with $\tau = 0.01$.

B.2 Conservative Q-learning implementation

The conservatism penalty in Section 4.2 requires estimating expectations under $\pi_\beta(a|s)$ and $\tilde{\pi}_\beta(a|s) \propto (1 - \pi_\beta(a|s))$, with the latter being especially non-trivial to estimate. We employ a simple and crude approximation that we found to work well in practice, replacing $\pi_\beta(a|s)$ with the empirical distribution corresponding, for each sampled state-action tuple $(s_j, a_j) \in \mathcal{D}$, to a Dirac delta centered on a_j, such that $\pi_\beta(a|s_j) = \delta(a = a_j)$. This results in a simple expression for $\tilde{\pi}_\beta(a|s_j)$

corresponding to the uniform distribution over all *other* actions, such that $\tilde{\pi}_\beta(a|s_j) \propto \delta(a \neq a_j)$. After discretizing the actions, there are $N-1$ bins per dimension to exhaustively iterate over when computing the conservatism term in Eq. 2, which is the same as taking the average over targets for all unseen action values. In our experiments, we find that simply setting the conservatism weight to $\alpha = 1.0$ worked best, without additional tuning.

B.3 Robot system overview

The robot that we use in this work is a mobile manipulator with a 7-DOF arm with a 2 jaw parallel gripper, attached to a mobile base with a head-mounted RGB camera, illustrated in Figure 1. The RGB camera provides a 640×512 RGB image, which is downsampled to 320×256 before being consumed by the Q-Transformer. See Figure 4 for images from the robot camera view. The learned policy is set up to control the arm and the gripper of the robot. Our action space consists of 8 dimensions: 3D position, 3D orientation, gripper closure command, and an additional dimension indicating that the episode should be terminated, which the policy must trigger to receive a positive reward upon successful task completion. Position and orientation are relative to the current pose, while the gripper command is the absolute close fraction, ranging from fully open to fully closed. Orientation is represented via axis-angles, and all actions except whether to terminate are continuous actions discretized over their full action range in 256 bins. The termination action is binary, but we pad it to be the same size as the other action dimensions to avoid any issues with unequal weights. The policy operates at 3 Hz, with actions executed asynchronously [73].

C Pseudo-code

Algorithm 1 shows the loss computation for training each action dimension of the Q-Transformer. We first use Eq. 1 to compute the maximum Q-values over the next action dimensions. Then we compute the Q-target for the given dataset action by using the Bellman update with an additional maximization over the Monte-Carlo return and predicted maximum Q-value at the next time step. The TD-error is then computed using the Mean-Squared Error. Finally, we set a target of 0 for all discretized action bins except the dataset action and add the averaged Mean-Squared Error over these dimensions to the TD-Error, which results in the total loss $\mathcal{L}$.

Algorithm 1 Temporal difference error and loss computation for one action dimension i at timestep t, a_t^i.

Input Sequence of state in time window of size w, $s_{t-w:t}$.
Input Language embedding of task instruction l.
Input The state at timestep $t+1$, s_{t+1}.
Input Dataset action up to dimension i, $\{_{\mathcal{D}}a_t^j\}_{j=0}^i$.
Output The loss to optimize Q-Transformer.

$Q^{targ} \leftarrow$ Compute maximum Q-values of the next action dimension using Eq. 1

`// Compute the maximum between Q-target and Monte Carlo return.`
$Q^{targ} \leftarrow \max(\text{MC}, Q^{targ})$

`// Compute the temporal difference error.`
$$\text{TDError} = \frac{1}{2}\left(\text{Q-Transformer}(l, s_{t-w:t}, \{a^j\}_{j=1}^i) - Q^{targ}\right)^2$$

```
// Compute the conservative regularizer.
// The sum is over all action bins not equal to the tokenized
   dataset action.
// N is the number of discretization bin.
```
$$\text{Reg} = \frac{1}{2(N-1)} \sum_{a \neq_{\mathcal{D}} a_t^i} \left(\text{Q-Transformer}(l, s_{t-w:t}, \{a^j\}_{j=1}^{i-1} \cup \{a\})\right)^2$$

`// Compute the loss function`
$\mathcal{L} = \text{TDError} + \text{Reg}$

Return $\mathcal{L}$ as the loss function to optimize Q-Transformer with.

D Q-Transformer value function with a language planner experiments

Recently, the SayCan algorithm [8] was proposed as a way to combine large language models (LLMs) with learned policies and value functions to solve long-horizon tasks. In this framework, the value function for each available skill is used to determine the "affordance" of the current state for that skill, and a large language model then selects from among the available affordances to take a step toward performing some temporally extended task. For example, if the robot is commanded to bring all the items on a table, the LLM might propose a variety of semantically meaningful items, and select from among them based on the item grasping skill that currently has a high value (corresponding to items that the robot thinks it can grasp). SayCan uses QT-Opt in combination with sim-to-real transfer to train Q-functions for these affordances. In the following set of experiments, we demonstrate that the Q-Transformer outperforms QT-Opt for affordance estimation without using any sim-to-real transfer, entirely using the real world dataset that we employ in the preceding experiments.

We first benchmark Q-Transformer on the problem of correctly estimating task affordances from the RT-1 dataset [1]. In addition to the standard training on demonstrations and autonomous data, we introduce a training with relabeling, which we found particularly useful for affordance estimation. During relabeling, we sample a random alternate task for a given episode. We relabel the task name of the episode to the newly sampled task, and set reward to 0.0. This ensures that the boundaries between tasks are more clearly learned during train-

Model	Precision	Recall	F1
QT-Opt (sim-to-real)	0.61	0.68	0.64
Q-T w/ relabel	0.76	0.89	**0.82**
Q-T w/o relabel	0.58	0.93	0.71

Table 1: Affordance estimation comparison: precision, recall and F1 score when using Q-values to determine if a task is feasible. Q-Transformer (Q-T) with multi-task relabeling consistently produces better affordance estimates.

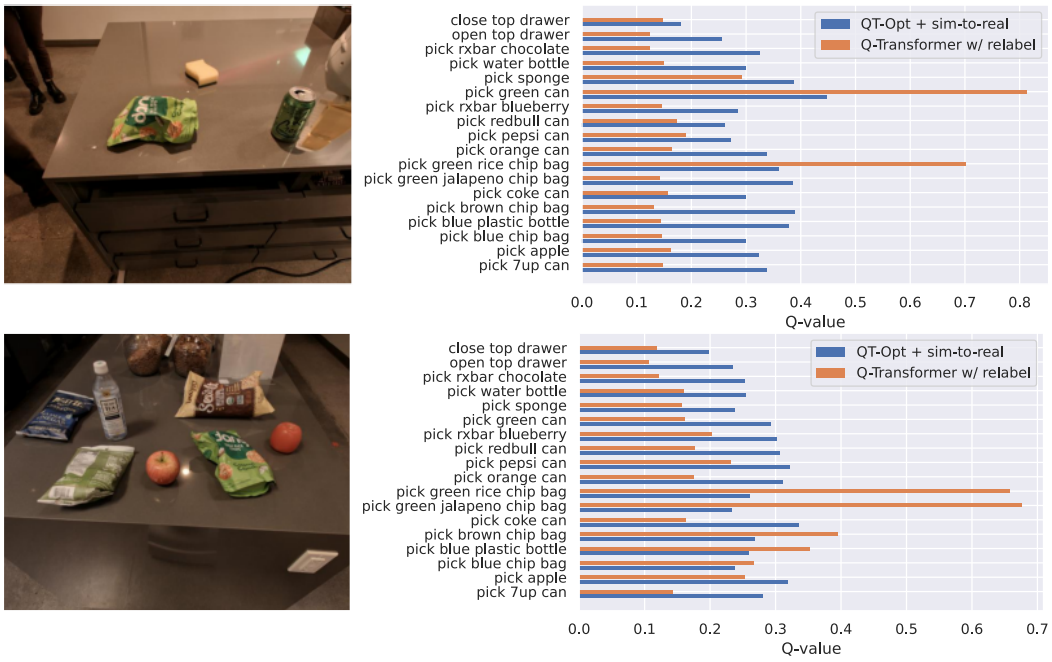

Figure 7: Qualitative comparisons of Q-values from QT-Opt (sim-to-real) and Q-Transformer. Q-Transformer outputs sharper Q-values for objects close to the robot, which can be grasped faster and more easily than far objects.

ing. Table 1 shows comparison of performance of our model with and without relabeling as well as the sim-to-real QT-Opt model used in SayCan [8]. Both of our models outperform the QT-Opt model on F1 score, with the relabeled model outperforming it by a large margin. This demonstrates that our Q-function can be effectively used for affordance estimation, even without training with sim-to-real transfer. Visualization of the Q-values produced by our Q-function can be found in Figure 7.

We then use Q-Transformer in a long horizon SayCan style evaluation, replacing both the sim-to-real QT-Opt model for affordance estimation, and the RT-1 policy for low-level robotic control. During this evaluation, a PaLM language model [74] is used to propose task candidates given a user query. Q-values are then used to pick the task candidate with the highest affordance score, which is then executed on the robot using the execution policy. The Q-Transformer used for affordance estimation is trained with relabeling. The Q-Transformer used for low-level control is trained without relabeling, since we found relabeling episodes at the task level did not improve execution performance. SayCan with Q-Transformer is better at both planning the sequence of tasks and executing those plans, as illustrated in Table 2.

Method		Success Rate	
Affordance	Execution	Planning	Execution
Q-T w/ relabel	Q-T	**93**	**93**
QT-Opt (sim-to-real)	RT-1	87	67

Table 2: Performance on SayCan style long-horizon tasks: SayCan queries $Q(s, a)$ in planning to pick a language instruction, then runs a policy to execute the plan. Q-Transformer outperforms RT-1 with QT-Opt in both planning and execution.

E Real robotic manipulation tasks used in our evaluation

We include the complete list of evaluation tasks in our real robot experiments below.

Drawer pick and place: pick 7up can from top drawer and place on counter, place 7up can into top drawer, pick brown chip bag from top drawer and place on counter, place brown chip bag into top drawer, pick orange can from top drawer and place on counter, place orange can into top drawer, pick coke can from middle drawer and place on counter, place coke can into middle drawer, pick orange

from middle drawer and place on counter, place orange into middle drawer, pick green rice chip bag from middle drawer and place on counter, place green rice chip bag into middle drawer, pick blue plastic bottle from bottom drawer and place on counter, place blue plastic bottle into bottom drawer, pick water bottle from bottom drawer and place on counter, place water bottle into bottom drawer, pick rxbar blueberry from bottom drawer and place on counter, place rxbar blueberry into bottom drawer.

Open and close drawer: open top drawer, close top drawer, open middle drawer, close middle drawer, open bottom drawer, close bottom drawer.

Move object near target: move 7up can near apple, move 7up can near blue chip bag, move apple near blue chip bag, move apple near 7up can, move blue chip bag near 7up can, move blue chip bag near apple, move blue plastic bottle near pepsi can, move blue plastic bottle near orange, move pepsi can near orange, move pepsi can near blue plastic bottle, move orange near blue plastic bottle, move orange near pepsi can, move redbull can near rxbar blueberry, move redbull can near water bottle, move rxbar blueberry near water bottle, move rxbar blueberry near redbull can, move water bottle near redbull can, move water bottle near rxbar blueberry, move brown chip bag near coke can, move brown chip bag near green can, move coke can near green can, move coke can near brown chip bag, move green can near brown chip bag, move green can near coke can, move green jalapeno chip bag near green rice chip bag, move green jalapeno chip bag near orange can, move green rice chip bag near orange can, move green rice chip bag near green jalapeno chip bag, move orange can near green jalapeno chip bag, move orange can near green rice chip bag, move redbull can near sponge, move sponge near water bottle, move sponge near redbull can, move water bottle near sponge, move 7up can near blue blastic bottle, move 7up can near green can, move blue plastic bottle near green can, move blue plastic bottle near 7up can, move green can near 7up can, move green can near blue plastic bottle, move apple near brown chip bag, move apple near green jalapeno chip bag, move brown chip bag near green jalapeno chip bag, move brown chip bag near apple, move green jalapeno chip bag near apple, move green jalapeno chip bag near brown chip bag.