A Summary of the Algorithm

Algorithm 1: Generative Skill Chaining (GSC) Algorithm

Hyperparameters:
Number of reverse diffusion steps T
Forward-backward dependency factor γ
Gradient score function weight α

Inputs:
Pre-defined skill library $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$
Individual skill diffusion score functions ϵ_π
Task skeleton $\Phi = \{\pi_0, \pi_1, \ldots, \pi_K\}$ be a sequence of skills of length K
Initial state $\mathbf{s}^{(0)}$
Goal condition g
Constraints $h(\{\mathbf{s}, \mathbf{a}\})$: suppose $\{\mathbf{s}, \mathbf{a}\} = [\mathbf{s}^{(1)}, \mathbf{a}^{(2)}, \mathbf{s}^{(K)}]$ be the nodes affected by constraint

Initial skeleton solution $\mathbf{x}_T = [\mathbf{s}_T^{(0)}, \mathbf{a}_T^{(0)}, \mathbf{s}_T^{(1)}, \mathbf{a}_T^{(1)}, \ldots, \mathbf{s}_T^{(K)}]$ sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$
Initialize $t = T$
while $t \geq 0$ **do**

  // Score of skeleton sequence
  $\epsilon_\Phi(\mathbf{s}_t^{(0)}, \mathbf{a}_t^{(0)}, \mathbf{s}_t^{(1)}, \mathbf{a}_t^{(1)}, \ldots, \mathbf{s}_t^{(K)}, t) = \mathbf{0}$
  for $i = 1 : K$ **do**
    // Update subvectors of ϵ_Φ
    $\epsilon_\Phi(\mathbf{s}_t^{(i-1)}, \mathbf{a}_t^{(i-1)}, \mathbf{s}_t^{(i)}, t) = \epsilon_\Phi(\mathbf{s}_t^{(i-1)}, \mathbf{a}_t^{(i-1)}, \mathbf{s}_t^{(i)}, t) + \epsilon_{\pi_i}(\mathbf{s}_t^{(i-1)}, \mathbf{a}_t^{(i-1)}, \mathbf{s}_t^{(i)}, t)$
    $\epsilon_\Phi(\mathbf{s}_t^{(i)}, t) = \epsilon_\Phi(\mathbf{s}_t^{(i)}, t) - \Big(\gamma\epsilon_{\pi_i}(\mathbf{s}_t^{(i)}, t) + (1-\gamma)\epsilon_{\pi_{i+1}}(\mathbf{s}_t^{(i)}, t)\Big)$
  end

  // Constraint handling
  for $\boldsymbol{\nu} \in [\boldsymbol{s}^{(1)}, \boldsymbol{a}^{(2)}, \boldsymbol{s}^{(K)}]$ **do**
    // Update subvectors of ϵ_Φ
    $\epsilon_\Phi(\mathbf{v}_t, t) = \epsilon_\Phi(\mathbf{v}_t, t) - \alpha\nabla_{\mathbf{v}_t} \log h(\tilde{\mathbf{s}}^{(1)}, \tilde{\mathbf{a}}^{(2)}, \tilde{\mathbf{s}}^{(K)})$
  end

  // Obtain denoised samples
  $\tilde{\mathbf{x}}_0 = \mathbf{x}_t + \sigma_t\epsilon_\Phi(\mathbf{s}_t^{(0)}, \mathbf{a}_t^{(0)}, \mathbf{s}_t^{(1)}, \mathbf{a}_t^{(1)}, \ldots, \mathbf{s}_t^{(K)}, t)$

  // Get the updated noisy samples
  $q_{0(t-1)}(\mathbf{x}_{t-1}|\tilde{\mathbf{x}}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mathbf{x}}_0, \sigma_{t-1}^2\mathbf{I})$
  $t = t - 1$
end
Return $\mathbf{x}_0$

Hyperparameters and Computation The number of reverse diffusion timesteps is an important parameters which plays a key role in deciding the time required to complete the sampling while keeping up with the quality of the generated samples. While a lower number of steps reduces the time taken for sampling, higher number of steps leads to finely denoised high-quality samples. We try with numerous values (256, 128, 64, and 50) and converge to using 128 diffusion steps for most of the tasks. The dependency factor γ is set to be 0.5 following the explanations described in section 4 and section 5 (Figure 6). A value of $\gamma = 1$ makes GSC the same as a trivial policy rollout approach. Finally, in case of gradients, we finetune the weights to balance the effect of constraints in the reverse diffusion process. While it is difficult to drastically change the sampling trajectory due to the intricacies of the reverse process, we use $\alpha = 1$ for all our tasks with given planning constraints.

Individual Diffusion Model Score Function Hyperparameters

We follow the score-network architecture of DiT [42] and adopt to their open-source implementation: `github.com/facebookresearch/DiT`. We use the following hyperparameters for building our score-network:

Table 3: Hyperparameters for Score-Network with Transformer Backbone

Hyper-parameter	Value
Hidden Dimension	128
Number of Blocks	4
Number of Heads	4
MLP Ratio	4
Dropout Probability	0.1
Number of Input Channels	17
Number of Output Channels	17

B Additional Discussion

Implementation details. The performance of the proposed skill sequencing framework depends on the diversity of the expert dataset, the maximum horizon dependency of action in the unseen skeleton, and the quality of the trained skill diffusion models. Furthermore, only the true distributions are estimated and used to sample candidate solutions. To ensure high-success probability for all our tasks, we consider sampling multiple candidate sequence solutions (two of them are shown in Figure 7) and consider the best probable solution based on the product of individual skill success probability metric.

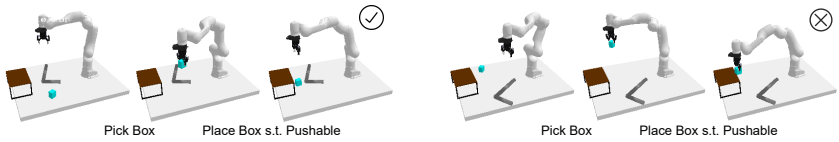

Figure 7: For a "pick-place" task of picking-and-placing the cyan box such that it can be pushed inside the rack: **Left** A correctly sampled state sequence while **Right** is incorrect. Hence, filtering candidate solutions is necessary.

Decision diffuser approach: state diffusion model with inverse dynamics actions. Diffusion models have been used for planning in robotics. One such framework is that of the decision diffuser [40], which samples a desired state trajectory and uses an inverse dynamics model to find the best action sequence. Our framework can achieve this by removing the action from the samples. However, this results in the distribution generated by diffusion models to be disjoint from the actions given by the inverse dynamics models. This distribution shift is sensitive to the quality of the sampled states and hence results in cascading errors. Considering a joint distribution of *state-action-state* transitions is advantageous as it is less sensitive to such state perturbabtions.

C Task Descriptions

As described in section 5, we evaluate our framework on three task domains (`hook reach`, `constrained packing`, and `rearrangement push`) with three tasks each. In addition, we validate the algorithm on a more complex skill with longer-horizon action dependency and describe it as the fourth task under the domain of `rearrangement push`. We describe all of such considered tasks below.

Hook Reach Task 1 sub-sequence of Figure 8

- **Scene:** Box is out of workspace, Hook is inside workspace
- **Goal:** Pick the Box
- **Skeleton: Pick** Hook, **Pull** Box, **Place** Hook, **Pick** Box

Hook Reach Task 2 easy version of Figure 8

- **Scene:** Yellow Box is out of workspace, Blue Box inside the workspace, Hook is inside workspace, Rack is inside workspace, Rack is empty
- **Goal:** Yellow Box on Rack
- **Skeleton: Pick** Hook, **Pull** Yellow Box, **Place** Hook, **Pick** Yellow Box, **Place** Yellow Box on Rack

Hook Reach Task 3 shown in Figure 8

- **Scene:** Red Box is out of workspace, Hook is inside workspace, Rack is inside workspace, Rack already has two blocks (Yellow and Blue)
- **Goal:** Red Box on Rack (without collision)
- **Skeleton: Pick** Hook, **Pull** Red Box, **Place** Hook, **Pick** Red Box, **Place** Red Box on Rack

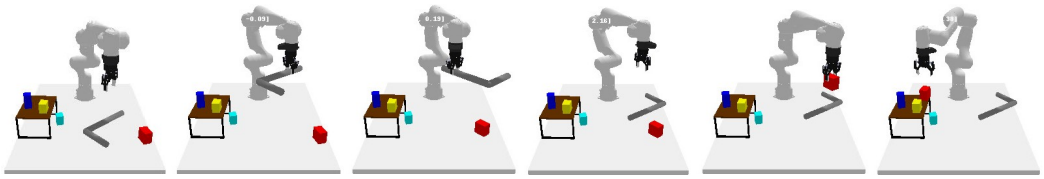

Figure 8: Hook Reach Task 3

Constrained Packing Task 1 shown in Figure 9

- **Scene:** Three boxes in the workspace, Rack is in workspace, Blue block on Rack
- **Goal:** All Boxes on Rack (without collision)
- **Skeleton: Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack

Constrained Packing Task 2 sub-sequence of Figure 10

- **Scene:** Three boxes in the workspace, Rack is in workspace, Rack is empty
- **Goal:** Three Boxes on Rack (without collision)
- **Skeleton: Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack

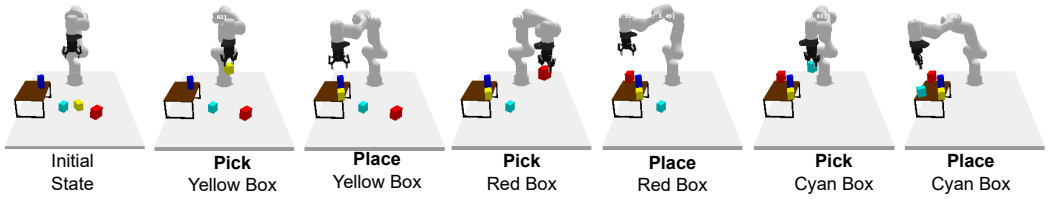

Figure 9: Constrained Packing Task 1

Constrained Packing Task 3 shown in Figure 10

- **Scene:** Four boxes in the workspace, Rack is in workspace, Rack is empty
- **Goal:** Four Boxes on Rack (without collision)
- **Skeleton: Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack, **Pick** Box, **Place** Box on Rack

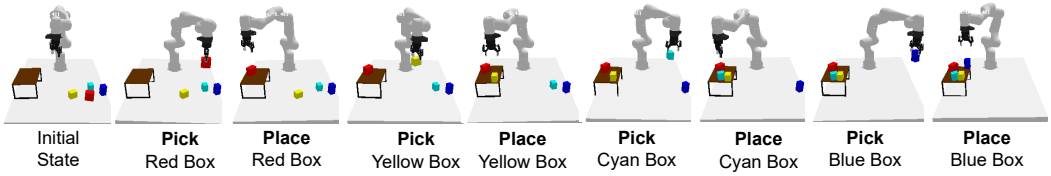

Figure 10: Constrained Packing Task 3

Rearrangement Push Task 1 shown in Figure 11

- **Scene:** Box in workspace, Hook in workspace, Rack outside workspace
- **Goal:** Box under Rack
- **Skeleton: Pick** Box, **Place** Box, **Pick** Hook, **Push** Box using Hook

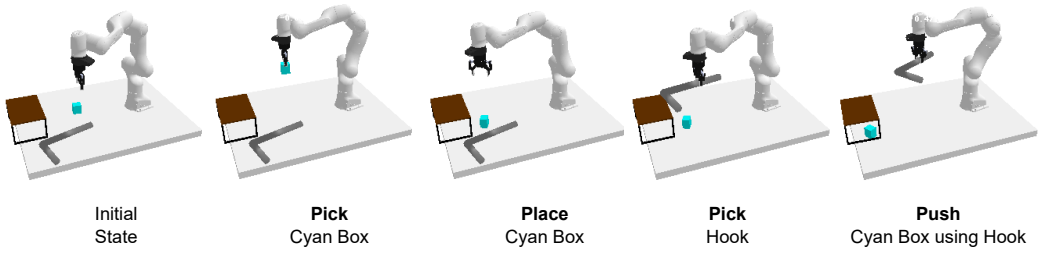

Figure 11: Rearrangement Push Task 1

Rearrangement Push Task 2 shown in Figure 12

- **Scene:** Three Boxes in workspace, Hook in workspace, Rack outside workspace
- **Goal:** Yellow Box under Rack
- **Skeleton: Pick** Hook, **Place** Hook, **Pick** Cyan Box, **Place** Cyan Box, **Pick** Hook, **Push** Yellow Box using Hook

Rearrangement Push Task 3 shown in Figure 13

- **Scene:** Four Boxes in workspace, Hook in workspace, Rack outside workspace
- **Goal:** Blue Box under Rack

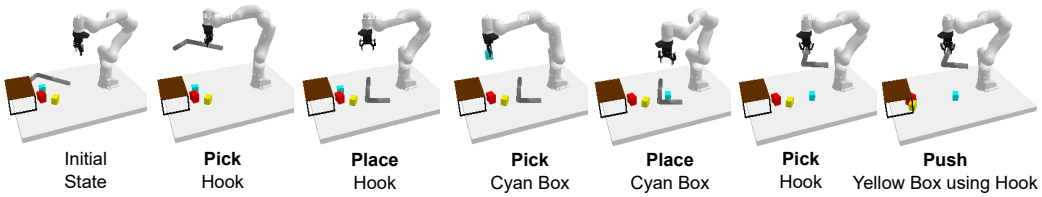

Figure 12: Rearrangement Push Task 2

- **Skeleton: Pick** Red Box, **Place** Red Box, **Pick** Yellow Box, **Place** Yellow Box, **Pick** Cyan Box, **Place** Cyan Box, **Pick** Hook, **Push** Blue Box using Hook

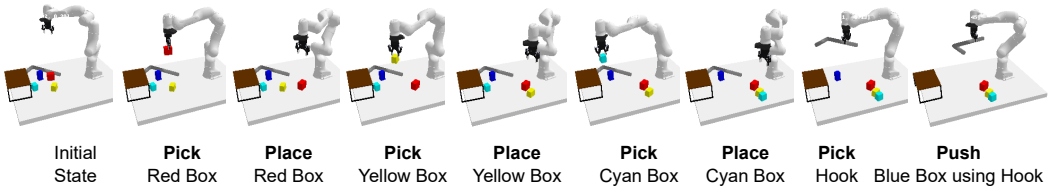

Figure 13: Rearrangement Push Task 3

Rearrangement Push Task 4 shown in Figure 14

- **Scene:** Box outside workspace, Hook in workspace, Rack outside workspace
- **Goal:** Box under Rack
- **Skeleton: Pick** Hook, **Pull** Box using Hook, **Place** Hook, **Pick** Box, **Place** Box, **Pick** Hook, **Push** Box using Hook

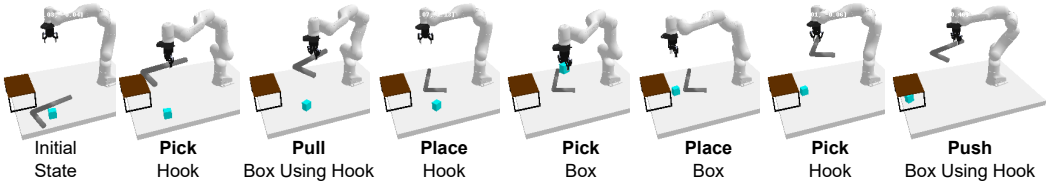

Figure 14: Rearrangement Push Task 4

D Additional Results

One of the attractive aspects of diffusion models is to visualize convergence to a valid solution starting from Gaussian noise. We visualize such results and show one of them below.

Figure 15: Reverse Diffusion visualization of `Rearrangement Push Task 3` for 50 timesteps.

E Hardware Experiment Setup

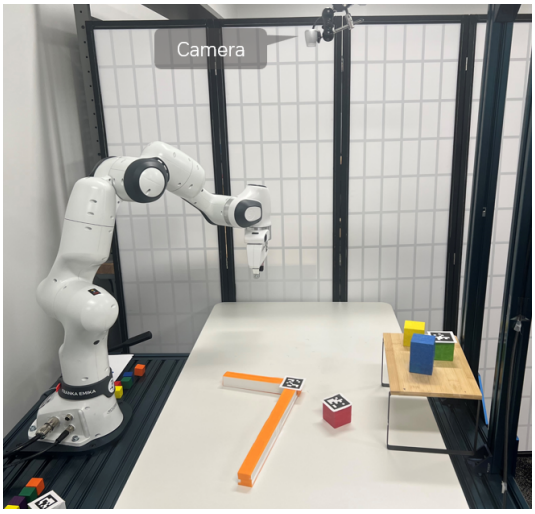

Figure 16: Hardware Experiment Setup

The experimental setup, illustrated in Fig. 16, encompasses a Franka Panda robot arm and an Intel RealscSense camera, several blocks, a rack, and a hock. The camera is positioned overhead, facing downward to fully observe the poses of all the objects. AprilTag is employed for SE(3) pose detection of the objects. During planning, the Frankx controller is utilized to generate smooth linear motion toward the desired gripper pose.