

417 Appendix

418 A Details for the Object Shape Inference Domain

419 Prior VAE Training Details

420 The prior used in this domain is a VAE trained on 2048-point PCs of ShapeNet “airplane” objects.
421 As it is based on the architecture used in Daniel and Tamar [27], we refer the reader to their work
422 for architectural details⁵. We train the VAE for 2000 iterations, augmenting the dataset with random
423 rotations around the vertical (z) axis in the range of $[-\frac{\pi}{4}, \frac{\pi}{4}]$. Both the encoder and decoder are
424 trained with the Adam optimizer [35], with a learning rate of 0.0005 and a batch size of 64. The
425 prior standard deviation is set to $\sigma_z = 0.2$, and the weighting parameters for the loss are set to
426 $\beta_{rec} = 50, \beta_{KL} = 1$. The latent space dimension is 128.

427 Grasping Simulator Implementation Details

428 To simplify implementation, we use a hand-crafted geometric simulator to calculate contact points
429 between a theoretical robot hand and object point clouds (PCs). We assume each finger is moved
430 along a vector pointing at the origin (which is located inside the object PC), and mark the point in
431 the PC furthest from the origin along this vector direction as the contact point. To simulate the width
432 of the finger, we consider points within a certain radius around the vector for contact calculation.
433 When grasping with k fingers, the observation $\mathbf{o} \in \mathbb{R}^{k \times 3}$ is the subset of contact points from the PC
434 \mathbf{x} .

435 Tuning Experiment Details

436 The prior $p(\mathbf{x}; \theta_0)$ is tuned for 2500 gradient steps, which takes approximately 25 seconds on a
437 single Nvidia GTX 1080 Ti GPU. We resample a new batch of $N = 256$ samples from the updated
438 model every $K = 32$ gradient steps, each taken on half of the batch due to memory constraints. We
439 use the Adam optimizer with learning rate 0.0002. We calculate the optimization objective with a
440 quantile value of $q = \frac{1}{16}$.

441 CVAE Baseline Hyperparameters

442 The CVAE baseline uses an architecture similar to the VAE prior model described above, with an
443 additional encoder to encode the condition contact points to a 128 dimensional latent $\mu_{prior}, \sigma_{prior}$.
444 In addition to its usage in the KL divergence loss, the prior mean μ_{prior} is injected into the decoder
445 in various layers. The CVAE baseline is mostly trained with the same hyperparameters as the VAE
446 described above, with two differences: $\beta_{KL} = 1$ and a learning rate of 0.0002.

447 Out-of-Distribution Experiment Visuals

448 Fig. 4 shows visual samples from the MACE-tuned prior and from the CVAE in the OOD experiment
449 (with the observation constituting a condition out of the distribution the CVAE was trained on).

450 Additional Model Samples

451 In this section we display additional samples for all of the distributions discussed in Sec. 4.1. All
452 visuals follow the same color scheme as in the main text, with samples shown in white and the grasp
453 positions represented by orange cylinders.

454 Fig. 5 shows samples from the pre-trained VAE prior. Figs. 6,7 display additional samples for the
455 first (in-distribution) experiment, for the posterior tuned by MACE and the CVAE baseline respec-
456 tively.

⁵See their public code at <https://github.com/taldatech/soft-intro-vae-pytorch>; we plan to release our code publicly at a later time.

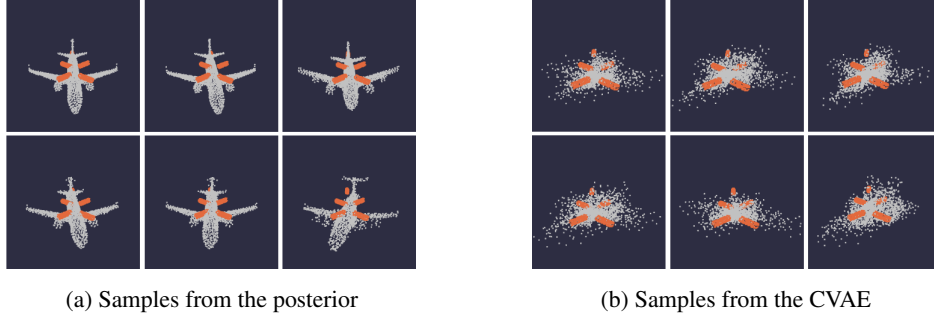


Figure 4: Samples from the posterior distribution tuned by MACE (left) and the CVAE baseline (right) when using an observation that is an OOD condition for the CVAE – note the gripper finger at the tail of the airplane. Results for MACE are similar to the in-distribution task, while the CVAE is unable to generate meaningful samples.

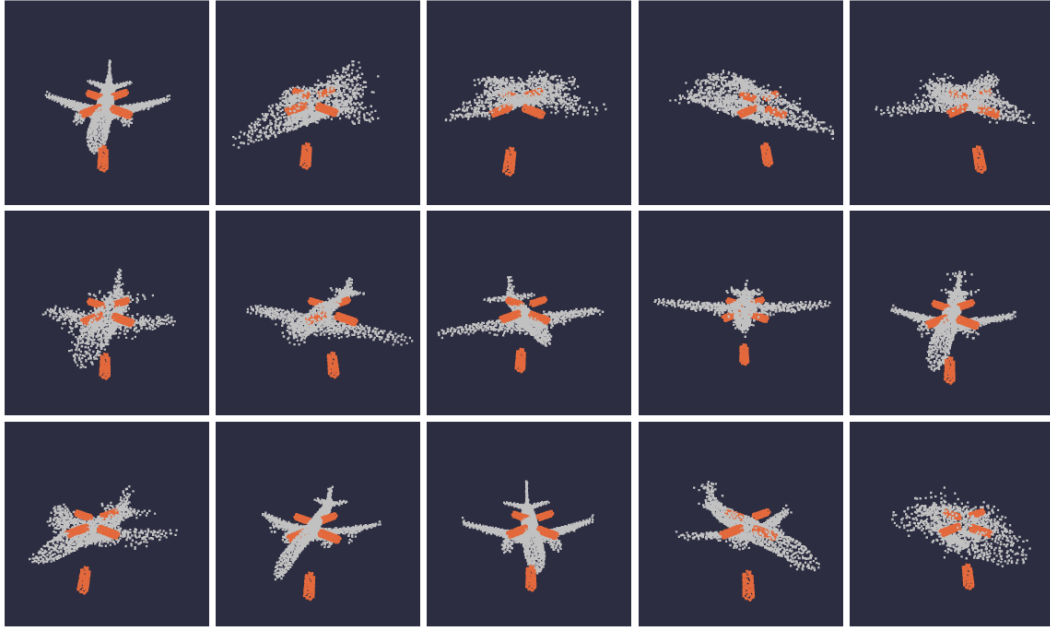


Figure 5: Samples from the prior

457 Figs. 8,9 display additional samples for the second (OOD) experiment, for the posterior tuned by
 458 MACE and the CVAE respectively.

459 B Details for the Inverse Kinematics Domain

460 Architecture of the Prior Model

461 As mentioned in Sec. 4.2, we train an autoregressive model to produce joint configurations condi-
 462 tioned on end-effector positions. We use 10M data points collected using the PyBullet simulator,
 463 and train the model end-to-end with the Adam optimizer in a supervised manner, using a maximum-
 464 likelihood objective over joint configurations.. Joint probabilities are represented by Gaussian mix-
 465 ture models with two components, each parameterized using a fully-connected NN with 5 layers of
 466 200 neurons, and Leaky ReLU activation functions.

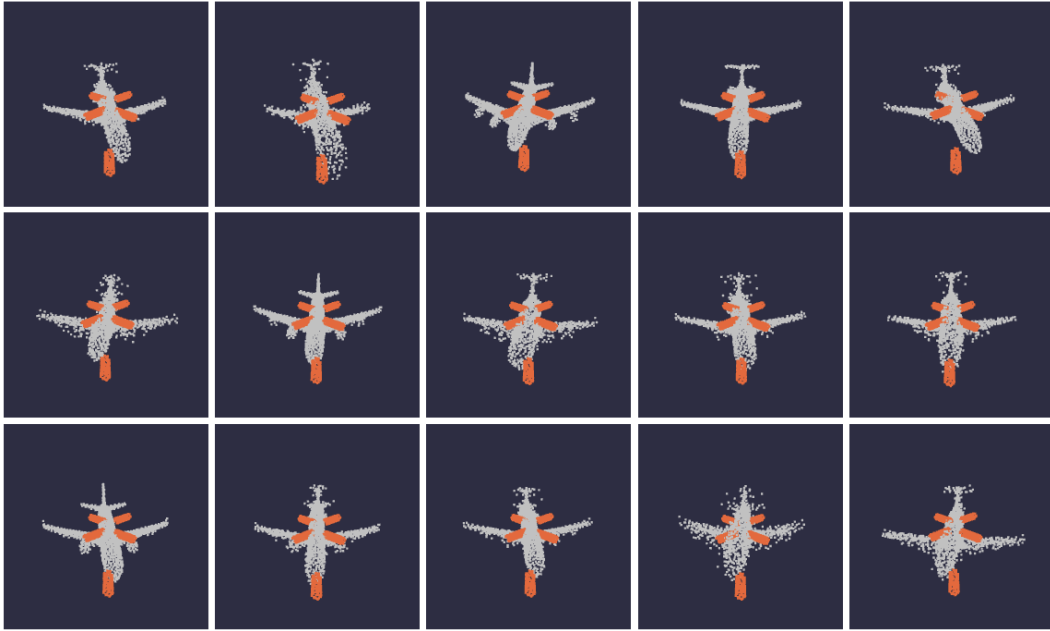


Figure 6: Samples from the posterior in the in-distribution experiment

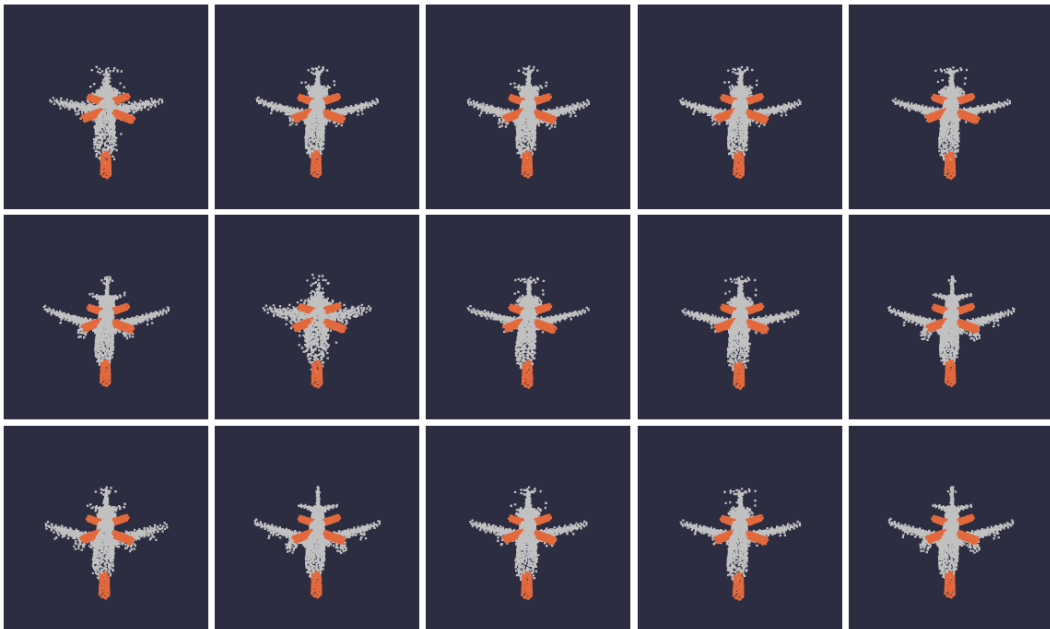


Figure 7: Samples from the CVAE in the in-distribution experiment

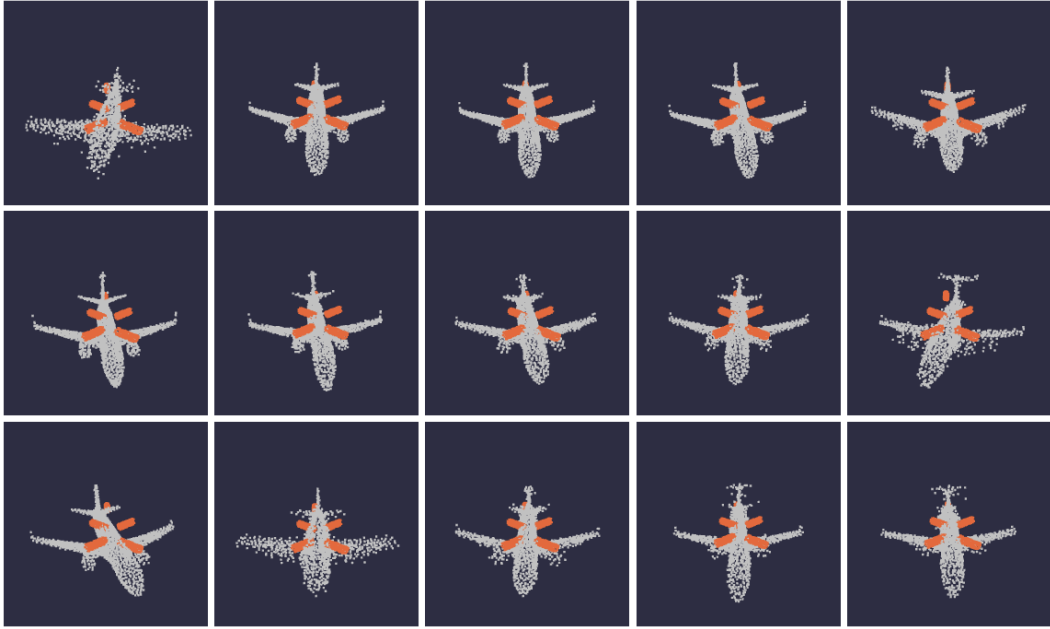


Figure 8: Samples from the posterior in the OOD experiment

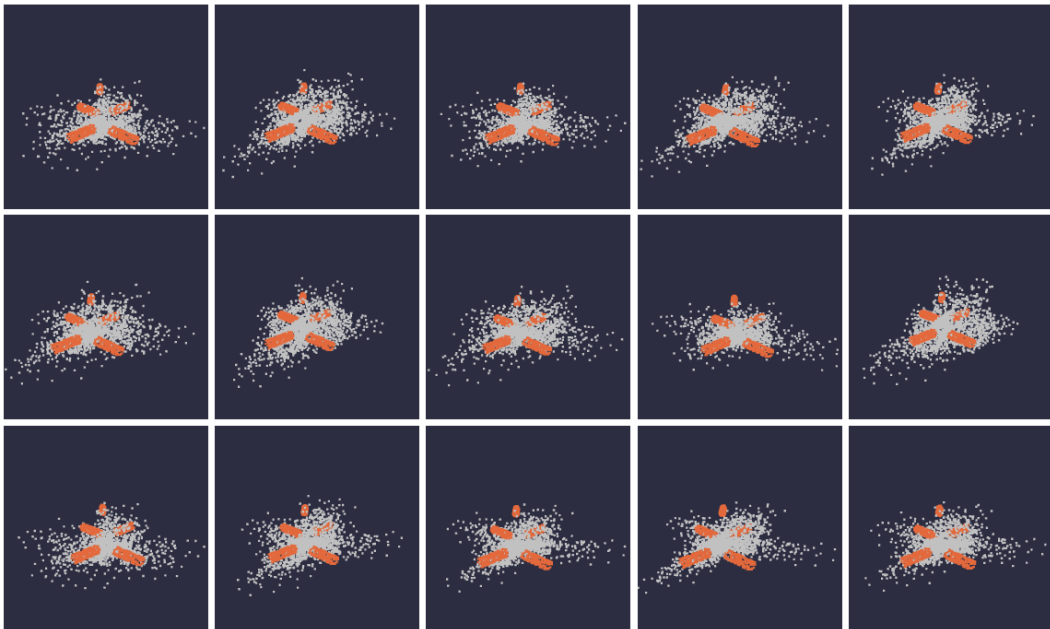


Figure 9: Samples from the CVAE in the OOD experiment

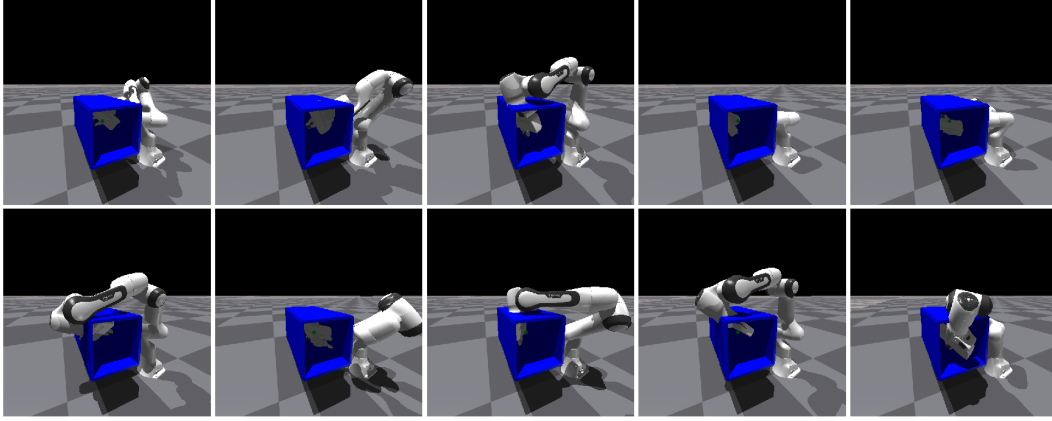


Figure 10: Samples from the prior overlaid with the box obstacle. Many of them collide with the walls of the box.

Experiment Details

PyBullet Experiments. The first two experiments described in Sec. 4.2 are conducted with the PyBullet physics simulation environment, with the wall and window obstacles. We tune the pre-trained prior for 1500 fine-tuning steps, which takes approximately 65 seconds on a single Nvidia GTX 1080 Ti GPU. We resample a new batch of $N = 64$ samples from the updated model every $K = 4$ gradient steps. We use the Adam optimizer with learning rate 0.00002. We calculate the optimization objective with a quantile value of $q = \frac{1}{16}$.

MoveIt and IsaacGym Experiment. For the box environment experiment and comparison to MoveIt, we use the same prior model, but instead use the GPU-based IsaacGym simulation environment to expedite scoring the samples. To calculate the results described in Table 2 of Sec. 4.2, we sample 20 batches of 4096 configurations each, and test them for collisions in IsaacGym. Obtaining the scores, we select the best configurations and report the mean and standard deviation of their distances from the goal in the accuracy column. The same configurations are used as initial positions for the “MACE + MoveIt” method in the third column, with the time constituting the total duration of sampling, testing for collisions with IsaacGym and finding solutions with MoveIt. The middle column reports times for MoveIt with a standard initial position. As MoveIt explicitly solves an optimization problem for the IK, its accuracy is very high; however, in some cases it takes much longer to find valid solutions.

Tuning Experiment for the Box Domain. In addition to the timing experiment, we conduct a tuning experiment with MACE on the box domain using IsaacGym. The experimental procedure is similar to the PyBullet experiments. We tune the model for 500 tuning steps, taking approximately 10 seconds on a single Nvidia GTX 1080 Ti GPU with the faster IsaacGym simulator. We resample a batch of $N = 4096$ configurations every $K = 4$ gradient steps, and use a quantile of $q = \frac{1}{128}$. Fine-tuning is conducted using the Adam optimizer, with a learning rate of 0.0001. Samples from the prior can be found in Fig. 10, while samples from the tuned model can be seen in Fig. 11.

Additional Model Samples for the PyBullet Experiments

In this section, we provide additional samples for the distributions described in Sec. 4.2. Fig. 12 and Fig. 14 provide samples from the prior model, trained with no obstacles present in the workspace. This is the same distribution in both sets of samples, overlaid with different objects to show that many configurations collide with each of them.

Fig. 13 shows samples from the posterior tuned with MACE in the presence of the wall obstacle. Fig. 15 shows samples from the posterior tuned with MACE and the window obstacle.

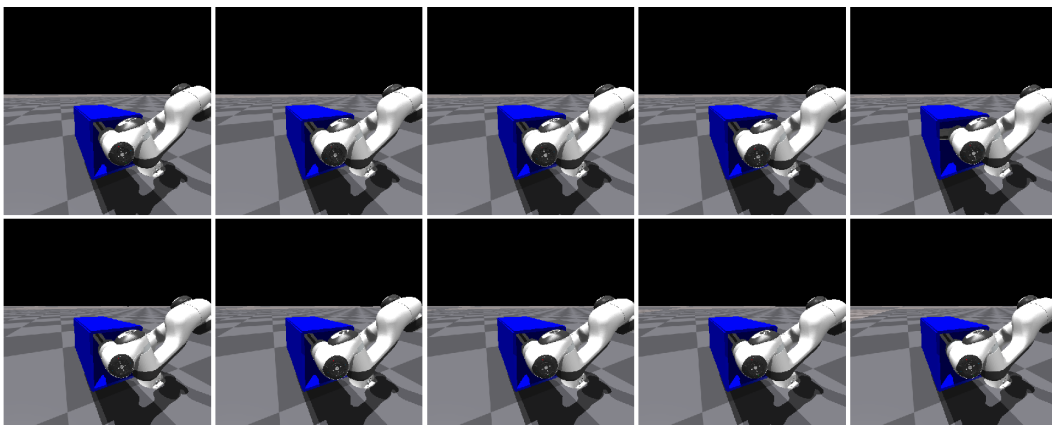


Figure 11: Samples from the posterior tuned with MACE to match the box obstacle.

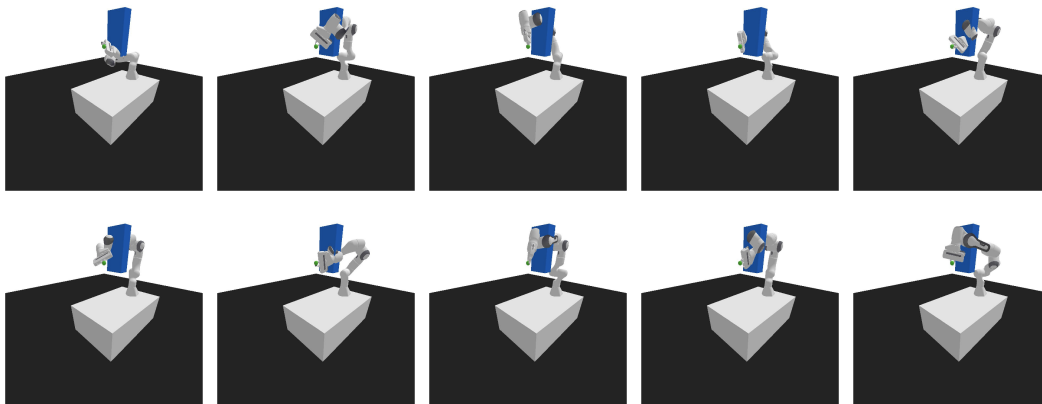


Figure 12: Samples from the prior overlaid with the wall obstacle.

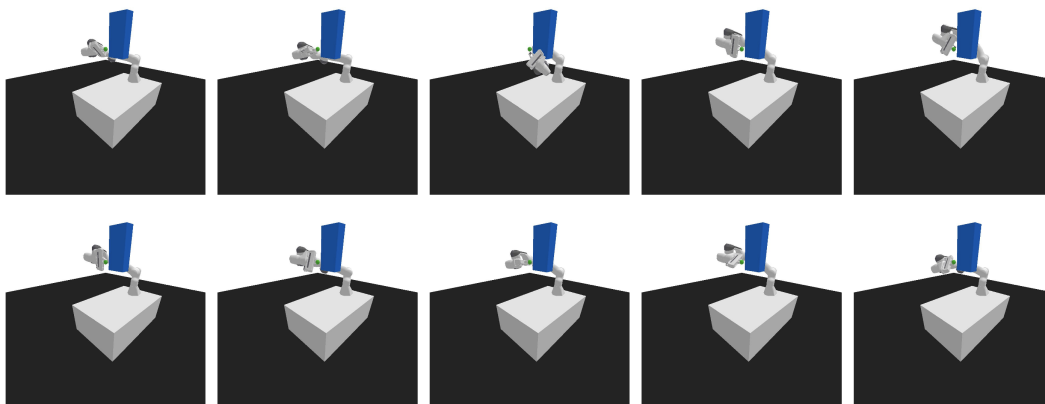


Figure 13: Samples from the posterior tuned to match observations of the wall obstacle.

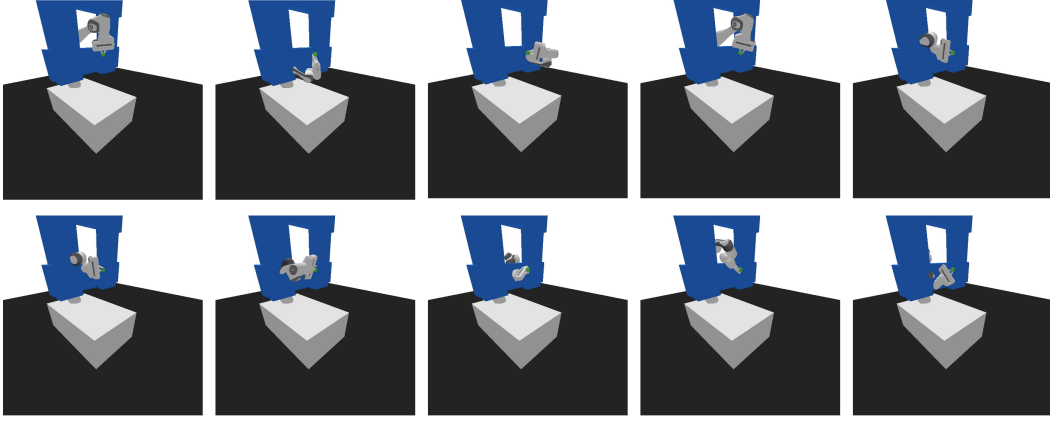


Figure 14: Samples from the prior overlaid with the window obstacle.

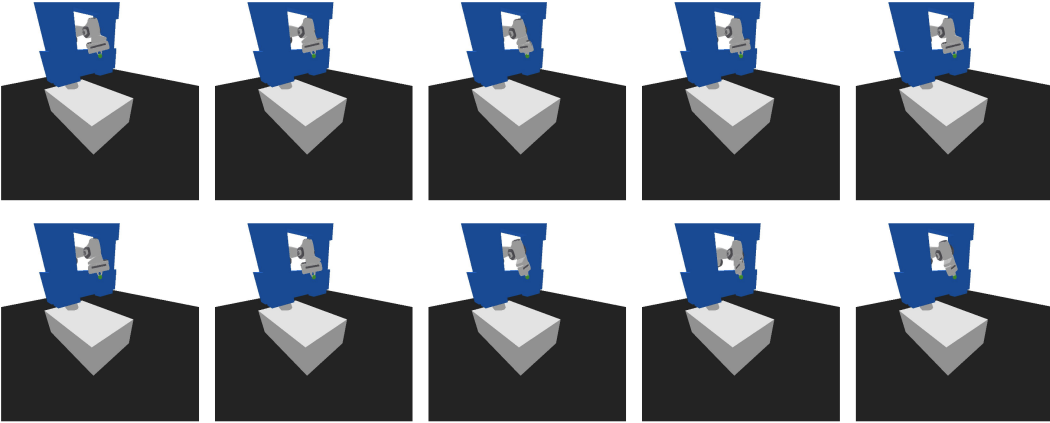


Figure 15: Samples from the posterior tuned to match observations of the window obstacle.

C The the Point Cloud Completion Domain

PC completion is an important component of manipulation pipelines, which allows robots to reason about their environment when partial information is available from sources such as depth sensors [36, 37]. Previous work typically focuses on scenarios in which a model can be faithfully recovered given the partial PC, i.e., when the dataset is small or the partial information is indicative of the object [38, 39, 40]. Instead, we consider a case where the posterior can be extremely multi-modal, and must therefore model a highly diverse distribution.

Given a partial PC as the observation \mathbf{o} , we infer a posterior distribution over possible full PCs \mathbf{x} . We include this domain as a proof-of-concept, and present qualitative results on a relatively simple dataset.

Dataset. We use a dataset of 10K symmetrical 3D boxes generated with random edge lengths, placed on the xy plane and centered around the z axis. Each PC consists of 2048 points, uniformly sampled on the box faces.

Model. We use a the same VAE architecture described in Sec. 4.1. The VAE is trained for 2000 iterations with training samples augmented by random rotation around the vertical (z) axis in the range of $[-\pi, \pi]$. Both the encoder and decoder are trained with the Adam optimizer [35], with a learning rate of 0.0002 and a batch size of 64. The prior standard deviation is set to $\sigma_z = 1$, and the weighting parameters for the loss are set to $\beta_{rec} = 1, \beta_{KL} = 0.1$. The latent space dimension is 128.

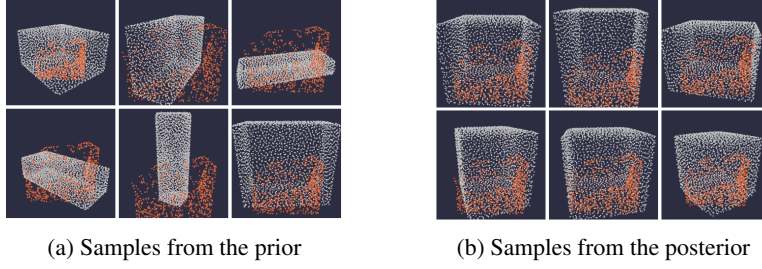


Figure 16: Tuning for the PC completion domain. Samples from the prior and posterior models are shown in white. Partial PC observation \mathbf{o} is overlaid over all samples in orange. While the prior model is extremely diverse and exhibits many different box sizes and rotations, the posterior tuned with MACE converges to samples which more closely match the evidence, while still producing a plausible distribution of objects.

Simulator. We require a simulator that can produce partial PCs given a full PC model. For this simple dataset, we obtain partial PCs by applying a random cut to each box, using a randomly sampled hyperplane. Note that this shape of the partial PC can fit a variety of different boxes, leading to a diverse posterior.

Score function. To measure similarity between PCs, $S(\mathbf{o}', \mathbf{o})$ is calculated using the Chamfer distance between PCs \mathbf{o}' and \mathbf{o} . As suggested by Chen et al. [37], we find that calculating the distance to the top $k > 1$ nearest points produces better results than $k = 1$, and therefore use $k = 5$ when calculating the score function. Considering PCs \mathbf{x} and \mathbf{x}' with points labeled as $\{p_i\}_{i=1}^N$ and $\{p'_i\}_{i=1}^M$ respectively, the original Chamfer distance is given by:

$$\text{CD} = \sum_{i=1}^N \min_{p'_i \in \mathbf{x}'} \|p'_i - p_i\|_2^2 + \sum_{i=1}^M \min_{p_i \in \mathbf{x}} \|p_i - p'_i\|_2^2.$$

The k -wise Chamfer distance replaces the min operation with a selection of the top- k nearest neighbors, denoted by the sets $\mathbf{x}^{(k)}$ and $\mathbf{x}'^{(k)}$:

$$\text{CD}_k = \frac{1}{k} \sum_{i=1}^N \sum_{p'_i \in \mathbf{x}'^{(k)}} \|p'_i - p_i\|_2^2 + \frac{1}{k} \sum_{i=1}^M \sum_{p_i \in \mathbf{x}^{(k)}} \|p_i - p'_i\|_2^2$$

To obtain scores in $[0, 1]$ with 1 being the maximum score, we set $S(\mathbf{o}', \mathbf{o}) = \exp(-\tau \text{CD}_k(\mathbf{o}', \mathbf{o}))$, where τ is a temperature parameter, set to $\tau = 0.1$ in our experiments.

Point Cloud Completion: Results

We use MACE-VAE (see Sec. 3.2.1) to tune the prior distribution parameters of the VAE latent space for 4000 fine-tuning steps, which take approximately 40 seconds on a single Nvidia GTX 1080 Ti GPU. We resample a new batch of $N = 256$ samples from the updated model every $K = 128$ gradient steps, each taken on a batch of half of the samples. We use the Adam optimizer with learning rate 0.001. We calculate the optimization objective with a quantile value of $q = \frac{1}{32}$. Fig. 16a shows samples from the prior distribution $p(\mathbf{x}; \theta_0)$ overlaid with the partial PC observation \mathbf{o} , while fig. 16b shows samples from the posterior model $P(\mathbf{x}; \theta_T)$ tuned with MACE. We observe that MACE can produce diverse completions of the partial PC. Additional samples can be found in Figures 17, 18.

Additional Model Samples

Fig. 17 shows samples from the pre-trained VAE prior. Fig. 18 displays additional samples for the posterior tuned by MACE.

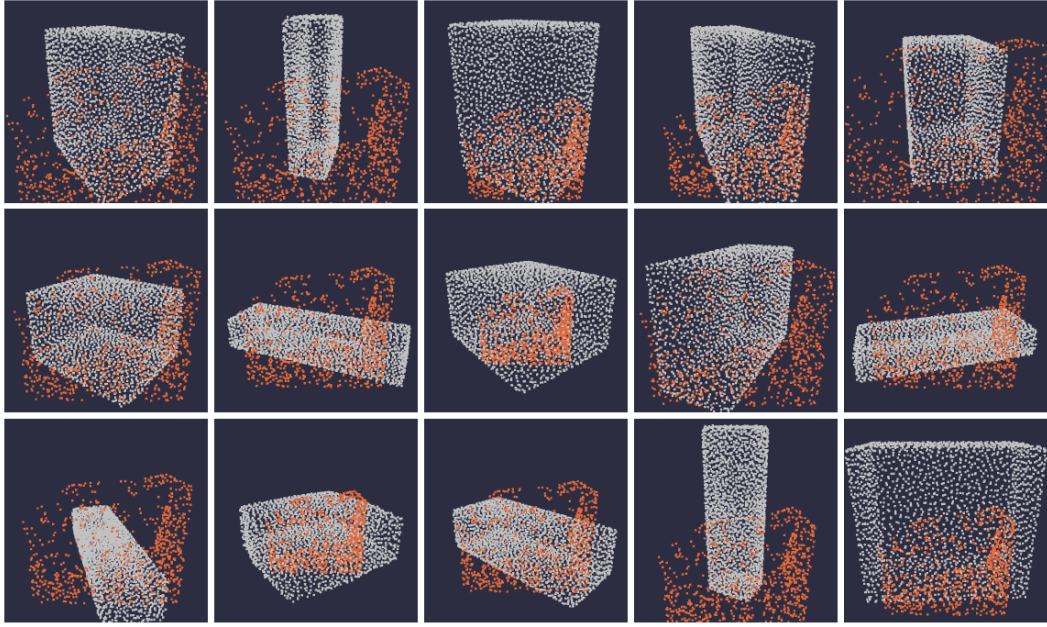


Figure 17: Samples from the prior

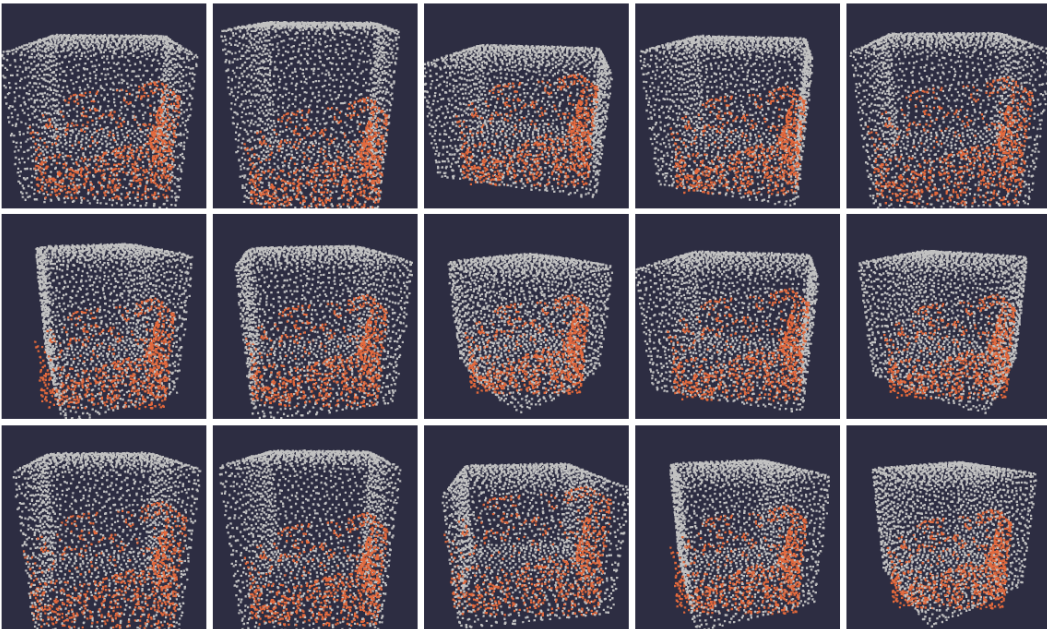


Figure 18: Samples from the posterior