A Proofs

A.1 Proof of Proposition 3.1

Expanding the entropy in Equation 5 we obtain

$$p^*(o) \propto \exp\big(\mathbb{E}_{p^*(\mathcal{D}|o)}[R_o(\mathcal{D})/\eta - \log p^*(\mathcal{D}|o)]\big).$$

Using $p^*(\mathcal{D}|o) = \tilde{p}(\mathcal{D}|o)/\sum_n \tilde{p}(\mathcal{D}_n|o)$ yields

$$\begin{aligned} p^*(o) \propto \exp\big(&\mathbb{E}_{p^*(\mathcal{D}|o)}[R_o(\mathcal{D})/\eta - \log \tilde{p}(\mathcal{D}|o) \\ &+ \log \sum_n \tilde{p}(\mathcal{D}_n|o)]\big). \end{aligned}$$

Next, leveraging that $\log \tilde{p}(\mathcal{D}|o) = R_o(\mathcal{D}_n)/\eta$ we see that

$$p^*(o) \propto \exp\big(\mathbb{E}_{p^*(\mathcal{D}|o)}[\log \sum_n \tilde{p}(\mathcal{D}_n|o)]\big) = \sum_n \tilde{p}(\mathcal{D}_n|o),$$

which concludes the proof. □

A.2 Proof of Corollary 3.1.2

We start by rewriting the lower bound as $L(\boldsymbol{\psi}, q) =$

$$\mathbb{E}_{p^*(o)}\big[\mathbb{E}_{p^*(\mathcal{D}|o)}[R_o(\mathcal{D}) - \eta \log p^*(\mathcal{D}|o)] - \eta \log p^*(o)\big].$$

Using $p^*(\mathcal{D}|o) \propto \tilde{p}(\mathcal{D}|o)$ and Proposition 3.1 we obtain

$$\begin{aligned} L(\boldsymbol{\psi}, q) =& \mathbb{E}_{p^*(o)}\big[\mathbb{E}_{p^*(\mathcal{D}|o)}[R_o(\mathcal{D}) - \eta \log \tilde{p}(\mathcal{D}|o) \\ &+ \eta \log \sum_n \tilde{p}(\mathcal{D}_n|o)] - \eta \log \sum_n \tilde{p}(\mathcal{D}_n|o) \\ &+ \eta \log \sum_o \sum_n \tilde{p}(\mathcal{D}_n|o)\big] \end{aligned}$$

With $\eta \log \tilde{p}(\mathcal{D}|o) = R_o(\mathcal{D}_n)$ all most terms cancel, giving

$$\begin{aligned} L(\boldsymbol{\psi}, q) =& \mathbb{E}_{p^*(o)}\big[\eta \log \sum_o \sum_n \tilde{p}(\mathcal{D}_n|o)\big] \\ =& \eta \log \sum_o \sum_n \tilde{p}(\mathcal{D}_n|o), \end{aligned}$$

which concludes the proof. □

A.3 Proof of Corollary 3.1.1

Expanding the expected KL divergence, we get

$$\begin{aligned} &\min_{\boldsymbol{\phi}} \mathbb{E}_{p(\mathcal{D})} D_{\mathrm{KL}}\big(p(o|\mathcal{D}) \| g_{\boldsymbol{\phi}}(o|\mathbf{x})\big) \\ =\ &\min_{\boldsymbol{\phi}} \sum_n p(\mathcal{D}_n) \sum_o p(o|\mathcal{D}_n) \log \frac{p(o|\mathcal{D}_n)}{g_{\boldsymbol{\phi}}(o|\mathbf{x}_n)}. \end{aligned}$$

Noting that $p(o|\mathcal{D}_n)$ is independent of $\boldsymbol{\phi}$ we can rewrite the objective as

$$\max_{\boldsymbol{\phi}} \sum_n p(\mathcal{D}_n) \sum_o p(o|\mathcal{D}_n) \log g_{\boldsymbol{\phi}}(o|\mathbf{x}_n).$$

Using that $p(o|\mathcal{D}) = \tilde{p}(\mathcal{D}|o)/\sum_o \tilde{p}(\mathcal{D}|o)$ together with $p(\mathcal{D}) = \sum_o p^*(o)p^*(\mathcal{D}|o)$ yields

$$\max_{\boldsymbol{\phi}} \sum_n \sum_o p^*(o)p^*(\mathcal{D}_n|o) \sum_o \frac{\tilde{p}(\mathcal{D}_n|o)}{\sum_o \tilde{p}(\mathcal{D}_n|o)} \log g_{\boldsymbol{\phi}}(o|\mathbf{x}_n).$$

Using Proposition 3.1 we can rewrite $p^*(o)p^*(\mathcal{D}|o)$ as $\tilde{p}(\mathcal{D}|o)/\sum_o\sum_n \tilde{p}(\mathcal{D}_n|o)$. Since the constant factor $1/\sum_o\sum_n \tilde{p}(\mathcal{D}_n|o)$ does not affect the optimal value of $\boldsymbol{\phi}$ we obtain

$$\max_{\boldsymbol{\phi}} \sum_n \sum_o \tilde{p}(\mathcal{D}_n|o) \sum_o \frac{\tilde{p}(\mathcal{D}_n|o)}{\sum_o \tilde{p}(\mathcal{D}_n|o)} \log g_{\boldsymbol{\phi}}(o|\mathbf{x}_n)$$
$$\max_{\boldsymbol{\phi}} \sum_n \sum_o \tilde{p}(\mathcal{D}_n|o) \log g_{\boldsymbol{\phi}}(o|\mathbf{x}_n),$$

which concludes the proof. $\square$

B Derivations

B.1 Lower Bound Decomposition

To arrive at Equation 4 by marginalizing over the latent variable o for the entropy of the joint curriculum, i.e.,

$$\begin{aligned}\mathcal{H}(\mathcal{D}) &= -\sum_n p(\mathcal{D}_n)\log p(\mathcal{D}_n)\\ &= -\sum_n p(\mathcal{D}_n)\sum_o p(o|\mathcal{D}_n)\log p(\mathcal{D}_n)\end{aligned}$$

Next, we use Bayes' theorem, that is, $p(\mathcal{D}_n) = p(o)p(\mathcal{D}_n|o)/p(o|\mathcal{D}_n)$, giving

$$\begin{aligned}\mathcal{H}(\mathcal{D}) = -\sum_n p(\mathcal{D}_n)\sum_o p(o|\mathcal{D}_n)\big(&\log p(o) + \log p(\mathcal{D}_n|o)\\ &- \log p(o|\mathcal{D}_n)\big).\end{aligned}$$

Moreover, we add and subtract the log auxiliary distribution $\log q(o|\mathcal{D}_n)$ which yields

$$\begin{aligned}\mathcal{H}(\mathcal{D}) = -\sum_n p(\mathcal{D}_n)\sum_o p(o|\mathcal{D}_n)\big(&\log p(o) + \log p(\mathcal{D}_n|o)\\ &- \log p(o|\mathcal{D}_n) + \log q(o|\mathcal{D}_n) - \log q(o|\mathcal{D}_n)\big).\end{aligned}$$

Rearranging the terms leads and writing the sums in terms of expectations we arrive at

$$\begin{aligned}\mathcal{H}(\mathcal{D}) =&\mathbb{E}_{p(o)}\big[\mathbb{E}_{p(o|\mathcal{D})}[\log q(o|\mathcal{D})] + \mathcal{H}(\mathcal{D}|o)\big] + \mathcal{H}(o)\\ &+ D_{\mathrm{KL}}\big(p(o|\mathcal{D})\|q(o|\mathcal{D})\big).\end{aligned}$$

Lastly, multiplying $\mathcal{H}(\mathcal{D})$ with η and adding $\mathbb{E}_{p(o)}\mathbb{E}_{p(\mathcal{D}|o)}[\log p_{\boldsymbol{\theta}_o}(\mathbf{y}|\mathbf{x},o)]$ we arrive at Equation 4 which concludes the derivation.

B.2 M-Step Objectives

Closed-Form Curriculum Updates. In order to derive the closed-form solution to Equation equation 5 (RHS) we solve

$$\max_{p(\mathcal{D}|z)} J_z(p(\mathcal{D}|z),\boldsymbol{\theta}_z) = \max_{p(\mathcal{D}|z)} \mathbb{E}_{p(\mathcal{D}|z)}[R_z(\mathcal{D})] + \eta\mathcal{H}(\mathcal{D}|z) \quad \text{subject to} \quad \sum_n p(\mathcal{D}) = 1.$$

Following the procedure of constrained optimization, we write down the Lagrangian function [55] as

$$\mathcal{L}(p,\lambda) = \sum_n p(\mathcal{D}_n|z)R_z(\mathcal{D}_n) - \eta\sum_n p(\mathcal{D}_n|z)\log p(\mathcal{D}_n|z) + \lambda(\sum_n p(\mathcal{D}_n|z) - 1),$$

where λ is the Lagrangian multiplier. As p is discrete, we solve for the optimal entries of $p(\mathcal{D}_n|z)$, that is, $p'(\mathcal{D}_n,\lambda|z) = \arg\max_p \mathcal{L}(p,\lambda)$. Setting the partial derivative of $\mathcal{L}(p,\lambda)$ with respect to p zero, i.e.,

$$\frac{\partial}{\partial p(\mathcal{D}_n|z)}\mathcal{L}(p,\lambda) = R_z(\mathcal{D}_n) - \eta\log p(\mathcal{D}_n|z) - \eta + \lambda \overset{!}{=} 0.$$

yields $p'(\mathcal{D}_n, \lambda|z) = \exp\big(R_z(\mathcal{D}_n) - \eta + \lambda\big)/\eta$.

Plugging p' back in the Lagrangian gives the dual function $g(\lambda)$, that is,

$$g(\eta) = \mathcal{L}(p', \lambda) = -\eta + \eta \sum_n \exp\big(R_z(\mathcal{D}_n) - \eta + \lambda\big)/\eta.$$

Solving for $\lambda^* = \arg\min_{\lambda \geq 0} g(\lambda)$ equates to

$$\frac{\partial}{\partial \lambda} g(\lambda) = -1 + \eta \sum_n \exp\big(R_z(\mathcal{D}_n) - \eta + \lambda\big)/\eta \overset{!}{=} 0$$
$$\iff \lambda^* = -\log\Big(\eta \sum_n \exp\big(R_z(\mathcal{D}_n) - \eta\big)/\eta\Big).$$

Finally, substituting λ^* into p' we have

$$p^*(\mathcal{D}_n|z) = p'(\mathcal{D}_n, \lambda^*|z) = \frac{\exp\big(R_z(\mathcal{D}_n)/\eta\big)}{\sum_n \exp\big(R_z(\mathcal{D}_n)/\eta\big)},$$

which concludes the derivation. The derivation of the optimal mixture weights $p^*(z)$ works analogously.

Expert Objective. In order to derive the expert objective of Equation 6 we solve

$$\max_{\boldsymbol{\theta}_z} J_z(p(\mathcal{D}|z), \boldsymbol{\theta}_z) = \max_{\boldsymbol{\theta}_z} \sum_n p(\mathcal{D}_n|z)\Big(\log p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z) + \eta \log q(z|\mathcal{D}_n) - \eta \log p(\mathcal{D}_n|z)\Big).$$

Noting that $q(z|\mathcal{D}_n)$ and $p(\mathcal{D}_n|z)$ are independent of $\boldsymbol{\theta}_z$ and $p(\mathcal{D}_n|z) = \tilde{p}(\mathcal{D}_n|z)/\sum_n \tilde{p}(\mathcal{D}_n|z)$ we find that

$$\max_{\boldsymbol{\theta}_z} J_z(p(\mathcal{D}|z), \boldsymbol{\theta}_z) = \max_{\boldsymbol{\theta}_z} \sum_n \frac{\tilde{p}(\mathcal{D}_n|z)}{\sum_n \tilde{p}(\mathcal{D}_n|z)} \log p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z).$$

Noting that $\sum_n \tilde{p}(\mathcal{D}_n|z)$ is a constant scaling factor concludes the derivation.

C Experiment Setup

C.1 Environments and Datasets

C.1.1 Obstacle Avoidance

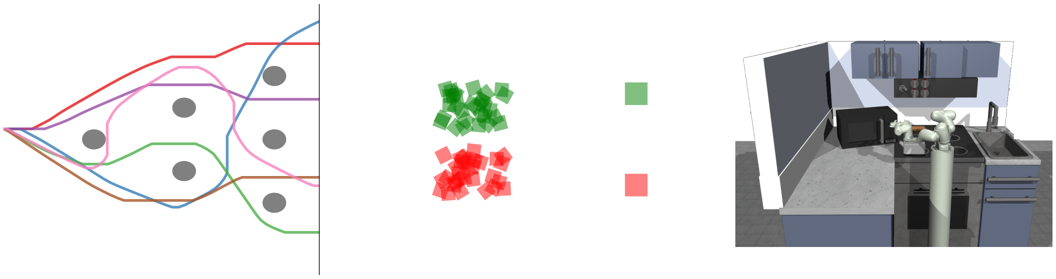


Figure 7: The left figure shows 6 out of 24 ways of completing the obstacle avoidance task. The middle figure shows 30 randomly sampled initial block configurations for the block pushing task. The right figure visualizes the Franka kitchen environment.

Dataset. The obstacle avoidance dataset contains 96 trajectories resulting in a total of 7.3k $(\mathbf{o}, \mathbf{a})$ pairs. The observations $\mathbf{o} \in \mathbb{R}^4$ contain the end-effector position and velocity in Cartesian space. Please note that the height of the robot is fixed. The actions $\mathbf{a} \in \mathbb{R}^2$ represent the desired position of the robot. The data is recorded such that there are an equal amount of trajectories for all 24 ways of avoiding the obstacles and reaching the target line. For successful example trajectories see Figure 7.

Performance Metrics. The *success rate* indicates the number of end-effector trajectories that successfully reach the target line (indicated by green color in Figure 3). The *entropy*

$$\mathcal{H}_{24}(\boldsymbol{\tau}) = -\sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau}) \log_{24} p(\boldsymbol{\tau}),$$

is computed for successful trajectories $\boldsymbol{\tau}$. To assess the model performance, we simulate 1000 end-effector trajectories. We count the number of successful trajectories for each way of completing the task. From that, we calculate a categorical distribution $p(\boldsymbol{\tau})$ which is used to compute the entropy. By the use of $\log_{24}$ we make sure that $\mathcal{H}_{24}(\boldsymbol{\tau}) \in [0, 1]$. If a model is able to discover all modes in the data distribution with equal probability, its entropy will be close to 1. In contrast, $\mathcal{H}_{24}(\boldsymbol{\tau}) = 0$ if a model only learns one solution.

C.1.2 Block Pushing

Dataset. The block pushing dataset contains 500 trajectories for each of the four push sequences (see Figure 4) resulting in a total of 2000 trajectories or 463k $(\mathbf{o}, \mathbf{a})$ pairs. The observations $\mathbf{o} \in \mathbb{R}^{16}$ contain the desired position and velocity of the robot in addition to the position and orientation of the green and red block. Please note that the orientation of the blocks is represented as quaternion number system and that the height of the robot is fixed. The actions $\mathbf{a} \in \mathbb{R}^2$ represent the desired position of the robot. This task is similar to the one proposed in [17]. However, they use a deterministic controller to record the data whereas we use human demonstrators which increases the difficulty of the task significantly due to the inherent versatility in human behavior.

Performance Metrics. The *success rate* indicates the number of end-effector trajectories $\boldsymbol{\tau}$ that successfully push both blocks to different target zones. To assess the model performance on non-successful trajectories, we consider the *distance error*, that is, the euclidean distance from the blocks to the target zones at the final block configuration of an end-effector trajectory. As there are a total of four push sequences (see Figure 3) we use the expected *entropy*

$$\mathbb{E}_{p(\mathbf{c})} \mathcal{H}_4(\boldsymbol{\tau}|\mathbf{c}) = -\sum_{\mathbf{c}} p(\mathbf{c}) \sum_{\boldsymbol{\tau}} p(\boldsymbol{\tau}|\mathbf{c}) \log_4 p(\boldsymbol{\tau}|\mathbf{c}),$$

to quantify a model's ability to cover the modes in the data distribution. Please note that we set $p(\mathbf{c}) = 1/30$ as we sample 30 block configurations uniformly from a configuration space (see Figure 7). For each $\mathbf{c}$ we simulate 16 end-effector trajectories. For a given configuration, we count how often each of the four push-sequences is executed successfully and use the result to calculate a categorical distribution $p(\boldsymbol{\tau}|\mathbf{c})$. Once repeated for all 30 configurations , we compute $\mathbb{E}_{p(\mathbf{c})} \mathcal{H}_4(\boldsymbol{\tau}|\mathbf{c})$. Using $\log_4$ we make sure that the expected entropy is upper bounded by 1. This bound is achieved if a model is able to execute each of the push sequences with equal probability for all configurations. If a model only executes one sequence successfully, the entropy is 0.

C.1.3 Franka Kitchen

Dataset. The Franka kitchen environment was introduced in [51]. It contains 566 human-collected trajectories resulting in a total of 128k $(\mathbf{o}, \mathbf{a})$ pairs. The observations $\mathbf{o} \in \mathbb{R}^{30}$ contain information about the position and orientation of the task-relevant objects in the environment. The actions $\mathbf{a} \in \mathbb{R}^9$ represent the signals to control the robot and the gripper. The dataset comprises sequences that successfully solve 4 out of 7 tasks in different orders.

Performance Metrics. First, we consider the *success rate* for a different number of tasks solved. We additionally compute the *entropy* over task sequences. This is computed using 100 simulated robot trajectories. For trajectories with a single task solved, we count how frequently each of the tasks is executed. From that, we calculate a categorical distribution which is then used for computing the entropy. We generalize this concept to more successful task completions, by calculating a categorical distribution over all 7^k possible task sequences for k task completions.

C.1.4 Table Tennis

Dataset. The table tennis dataset contains 5000 $(\mathbf{o}, \mathbf{a})$ pairs. The observations $\mathbf{o} \in \mathbb{R}^4$ contain the coordinates of the initial and target ball position as projection on the table. Movement primitives (MPs) [30] are used to describe the joint space trajectories of the robot manipulator using two basis functions per joint and thus $\mathbf{a} \in \mathbb{R}^{14}$.

Metrics. To evaluate the different algorithms on the demonstrations recorded using the table tennis environment quantitatively, we employ two performance metrics: The *success rate* and the *distance error*. The success rate is the percentage of strikes where the ball is successfully returned to the opponent's side. The distance error, is the distance between the target position and landing position of the ball for successful strikes.

C.1.5 Human Subjects for Data Collection

For the obstacle avoidance as well as the block pushing experiments we used data collected by humans. We note that all human subjects included in the data collection process are individuals who are collaborating on this work. The participants did, therefore, not receive any financial compensation for their involvement in the study.

C.2 IMC Details and Hyperparameter

IMC employs a parameterized inference network and conditional Gaussian distributions to represent experts. For the latter, we use a fixed variance of 1 and parameterize the means as neural networks. For both inference network and expert means we use residual MLPs [56]. For all experiments, we use batch-size $|\mathcal{B}| = |\mathcal{D}|$, number of components $N_z = 50$ and expert learning rate equal to 5×10^{-4}. Furthermore, we initialized all curriculum weights as $p(\mathcal{D}_n|z) = 1$. For the table tennis and obstacle avoidance task, we found the best results using a multi-head expert parameterization (see Section E.1) where we tested $1 - 4$ layer neural networks. We found that using 1 layer with 32 neurons performs best on the table tennis task and 2 layer with 64 neurons for the obstacle avoidance task. For the block pushing and Franka kitchen experiments, we obtained the best results using a sigle-head parameterization of the experts. We used 6 layer MLPs with 128 neurons for both tasks. For the inference network, we used a fixed set of parameters that are listed in Table 2. For the entropy scaling factor η we performed a hyperparameter sweep using Bayesian optimization. The respective values are $\eta = 1/30$ for obstacle avoidance, $\eta = 2$ for block pushing and Franka kitchen and $\eta = 1$ for table tennis.

Table 2: **IMC & EM Hyperparameter.**

PARAMETER	VALUE
EXPERT LEARNING RATE	10^{-4}
EXPERT BATCHSIZE	1024
EXPERT VARIANCE (σ^2)	1
INFERENCE NET HIDDEN LAYER	6
INFERENCE NET HIDDEN UNITS	256
INFERENCE NET EPOCHS	800
INFERENCE NET LEARNING RATE	10^{-3}
INFERENCE NET BATCHSIZE	1024

C.3 Baselines and Hyperparameter

We now briefly mention the baselines and their hyperparameters. We used Bayesian optimization to tune the most important hyperparameters.

Mixture of Experts trained with Expectation-Maximization (EM). The architecture of the mixture of experts model trained with EM [48] is identical to the one optimized with IMC: We employ a parameterized inference network and conditional Gaussian distributions to represent experts with the same hyperparameters as shown in Table 2. Furthermore, we initialized all responsibilities as $p(z|\mathbf{o}) = 1/N_z$, where N_z is the number of components.

Mixture Density Network (MDN). The mixture density network [8] uses a shared backbone neural network with multiple heads for predicting component indices as well as the expert likelihood. For the experts, we employ conditional Gaussians with a fixed variance. The model likelihood is maximized in an end-to-end fashion using stochastic gradient ascent. We experimented with different backbones and expert architectures. However, we found that the MDN is not able to partition the input space in a meaningful way, often resulting in sub-optimal outcomes, presumably due to mode averaging. To find an appropriate model complexity we tested up to 50 expert heads. We found that the number of experts heads did not significantly influence the results, further indicating that the MDN is not able to utilize multiple experts to solve sub-tasks. We additionally experimented with a version of the MDN that adds an entropy bonus to the objective [57] to encourage more diverse and multimodal solutions. However, we did not find significant improvements compared to the standard version of the MDN. For a list of hyperparameter choices see 3.

Table 3: **MDN Hyperparameter.** The 'Value' column indicates sweep values for the obstacle avoidance task, the block pushing task, the Franka kitchen task and the table tennis task (in this order).

Parameter	Sweep	Value
Expert hidden layer	$\{1, 2\}$	$1, 1, 1, 1$
Expert hidden units	$\{30, 50\}$	$50, 30, 30, 50$
Backbone hid. layer	$\{2, 3, 4, 6, 8, 10\}$	$3, 2, 4, 3$
Backbone hid. units	$\{50, 100, 150, 200\}$	$200, 200, 200, 200$
Learning rate $\times 10^{-3}$	$[0.1, 1]$	$5.949, 7.748, 1.299, 2.577$
Expert variance (σ^2)	–	1
Max. epochs	–	2000
Batchsize	–	512

Denoising Diffusion Probabilistic Models (DDPM). We consider the denoising diffusion probabilistic model proposed by [24]. Following common practice we parameterize the model as residual MLP [27] with a sinusoidal positional encoding [54] for the diffusion steps. Moreover, we use the cosine-based variance scheduler proposed by [58]. For further details on hyperparameter choices see Table 4.

Table 4: **DDPM Hyperparameter.** The 'Value' column indicates sweep values for the obstacle avoidance task, the block pushing task, the Franka kitchen task, and the table tennis task (in this order).

Parameter	Sweep	Value
Hidden layer	$\{4, 6, 8, 10, 12\}$	$6, 6, 8, 6$
Hidden units	$\{50, 100, 150, 200\}$	$200, 150, 200, 200$
Diffusion steps	$\{5, 15, 25, 50\}$	$15, 15, 15, 15$
Variance scheduler	–	cosine
Learning rate	–	10^{-3}
Max. epochs	–	2000
Batchsize	–	512

Normalizing Flow (NF). For all experiments, we build the normalizing flow by stacking masked autoregressive flows [59] paired with permutation layers [18]. As base distribution, we use a conditional isotropic Gaussian. Following common practice, we optimize the model parameters by maximizing its likelihood. See Table 5 for a list of hyperparameters.

Table 5: **NF Hyperparameter.** The 'Value' column indicates sweep values for the obstacle avoidance task, the block pushing task, the Franka kitchen task and the table tennis task (in this order).

Parameter	Sweep	Value
Num. flows	$\{4, 6, 8, 10, 12\}$	$6, 6, 4, 4$
Hidden units per flow	$\{50, 100, 150, 200\}$	$100, 150, 200, 150$
Learning rate $\times 10^{-4}$	$[0.01, 10]$	$7.43, 4.5, 4.62, 7.67$
Max. epochs	–	2000
Batchsize	–	512

Conditional Variational Autoencoder (CVAE). We consider the conditional version of the autoencoder proposed in [20]. We parameterize the encoder and decoder with a neural network with mirrored architecture. Moreover, we consider an additional scaling factor (β) for the KL regularization in the lower bound objective of the VAE as suggested in [60].

Table 6: **CVAE Hyperparameter.** The 'Value' column indicates sweep values for the obstacle avoidance task, the block pushing task, the Franka kitchen task and the table tennis task (in this order).

Parameter	Sweep	Value
Hidden layer	$\{4, 6, 8, 10, 12\}$	$8, 10, 4, 4$
Hidden units	$\{50, 100, 150, 200\}$	$100, 150, 100, 100$
Latent dimension	$\{4, 16, 32, 64\}$	$32, 16, 16, 16$
D_{KL} scaling (β)	$[10^{-3}, 10^{2}]$	$1.641, 1.008, 0.452, 0.698$
Learning rate	–	10^{-3}
Max. epochs	–	2000
Batchsize	–	512

Implicit Behavior Cloning (IBC). IBC was proposed in [17] and uses energy-based models to learn a joint distribution over inputs and targets. Following common practice we parameterize the model as neural network. Moreover, we use the version that adds a gradient penalty to the InfoNCE loss [17]. For sampling, we use gradient-based Langevin MCMC [56]. Despite our effort, we could not achieve good results with IBC. A list of hyperparameters is shown in Table 7.

Table 7: **IBC Hyperparameter.** The 'Value' column indicates sweep values for the obstacle avoidance task and the table tennis task (in this order). We do not get any good results for the block push task and the Franka kitchen task.

Parameter	Sweep	Value
Hidden dim	$\{50, 100, 150, 200, 256\}$	$200, 256$
Hidden layers	$\{4, 6, 8, 10\}$	$4, 6$
Noise scale	$[0.1, 0.5]$	$0.1662, 0.1$
Train samples	$[8, 64]$	$44, 8$
Noise shrink	–	0.5
Train iterations	–	20
Inference iterations	–	40
Learning rate	–	10^{-4}
Batch size	–	512
Epochs	–	1000

Behavior Transformer (BET). Behavior transformers were recently proposed in [22]. The model employs a minGPT transformer [61] to predict targets by decomposing them into cluster centers and residual offsets. To obtain a fair comparison, we compare our method to the version with no history. A comprehensive list of hyperparameters is shown in Table 8.

D Connection to Expectation Maximization

In this section we want to highlight the commonalities and differences between our algorithm and the expectation-maximization (EM) algorithm for mixtures of experts. First, we look at the updates of the variational distribution q. Next, we compare the expert optimization. Lastly, we take a closer look at the optimization of the gating distribution.

The EM algorithm sets the variational distribution during the E-step to

$$q(z|\mathbf{o}_n) = p(z|\mathbf{o}_n, \mathbf{a}_n) = \frac{p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z)p(z|\mathbf{o}_n)}{\sum_z p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z)p(z|\mathbf{o}_n)}, \tag{7}$$

Table 8: **BET Hyperparameter.** The 'Value' column indicates sweep values for the obstacle avoidance task, the block pushing task, the Franka kitchen task and the table tennis task (in this order).

PARAMETER	SWEEP	VALUE
TRANSFORMER BLOCKS	$\{2, 3, 4, 6\}$	$3, 4, 6, 2$
OFFSET LOSS SCALE	$\{1.0, 100.0, 1000.0\}$	$1.0, 1.0, 1.0, 1.0$
EMBEDDING WIDTH	$\{48, 72, 96, 120\}$	$96, 72, 120, 48$
NUMBER OF BINS	$\{8, 10, 16, 32, 50, 64\}$	$50, 10, 64, 64$
ATTENTION HEADS	$\{4, 6\}$	$4, 4, 6, 4$
CONTEXT SIZE	–	1
TRAINING EPOCHS	–	500
BATCH SIZE	–	512
LEARNING RATE	–	10^{-4}

for all samples n and components z. In the M-step, the gating distribution $p(z|\mathbf{o})$ is updated such that the KL divergence between $q(z|\mathbf{o})$ and $p(z|\mathbf{o})$ is minimized. Using the properties of the KL divergence, we obtain a global optimum by setting $p(z|\mathbf{o}_n) = q(z|\mathbf{o}_n)$ for all n and all z. This allows us to rewrite Equation 7 using the recursion in q, giving

$$q(z|\mathbf{o}_n)^{(i+1)} = \frac{p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z)q(z|\mathbf{o}_n)^{(i)}}{\sum_z p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z)q(z|\mathbf{o}_n)^{(i)}},$$

where (i) denotes the iteration of the EM algorithm. The update for the variational distribution of the IMC algorithm is given by

$$\begin{aligned} q(z|\mathcal{D}_n)^{(i+1)} &= \frac{\tilde{p}(\mathcal{D}_n|z)^{(i+1)}}{\sum_z \tilde{p}(\mathcal{D}_n|z)^{(i+1)}} \\ &= \frac{p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z)^{1/\eta}q(z|\mathcal{D}_n)^{(i)}}{\sum_z p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z)^{1/\eta}q(z|\mathcal{D}_n)^{(i)}}. \end{aligned}$$

Consequently, we see that $q(z|\mathbf{o}) = q(z|\mathcal{D})$ for $\eta = 1$. However, the two algorithms mainly differ in the M-step for the experts: The EM algorithm uses the variational distribution to assign weights to samples, i.e.

$$\max_{\boldsymbol{\theta}_z} \sum_{n=1}^{N} q(z|\mathbf{o}_n) \log p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z),$$

whereas IMC uses the curricula as weights, that is,

$$\max_{\boldsymbol{\theta}_z} \sum_{n=1}^{N} p(\mathcal{D}_n|z) \log p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z).$$

This subtle difference shows the properties of moment and information projection: In the EM algorithm each sample $\mathbf{o}_n$ contributes to the expert optimization as $\sum_z q(z|\mathbf{o}_n) = 1$. However, if all curricula ignore the nth sample, it will not have impact on the expert optimization. Assuming that the curricula ignore samples that the corresponding experts are not able to represent, IMC prevents experts from having to average over 'too hard' samples. Furthermore, this results in reduced outlier sensitivity as they are likely to be ignored for expert optimization. Lastly, we highlight the difference between the gating optimization: Assuming that both algorithms train a gating network $g_{\boldsymbol{\phi}}(z|\mathbf{o})$ we have

$$\max_{\boldsymbol{\phi}} \sum_n \sum_z q(z|\mathbf{o}_n) \log g_{\boldsymbol{\phi}}(z|\mathbf{o}_n),$$

for the EM algorithm and

$$\max_{\boldsymbol{\phi}} \sum_n \sum_z \tilde{p}(\mathcal{D}_n|z) \log g_{\boldsymbol{\phi}}(z|\mathbf{o}_n),$$

for IMC. Similar to the expert optimization, EM includes all samples to fit the parameters of the gating network, whereas IMC ignores samples where the unnormalized curriculum weights $\tilde{p}(\mathcal{D}_n|z)$ are zero for all components.

E Algorithm Details & Ablation Studies

E.1 Expert Design Choices

Distribution. In our mixture of experts policy, we employ Gaussian distributions with a fixed variance to represent the individual experts. This choice offers several benefits in terms of likelihood calculation, optimization and ease of sampling:

To perform the M-Step for the curricula (Section 3.3), exact log-likelihood computation is necessary. This computation becomes straightforward when using Gaussian distributions. Additionally, when Gaussian distributions with fixed variances are employed to represent the experts, the M-Step for the experts simplifies to a weighted squared-error minimization. Specifically, maximizing the weighted likelihood reduces to minimizing the weighted squared error between the predicted actions and the actual actions.

The optimization problem for the expert update can be formulated as follows:

$$\boldsymbol{\theta}_z^* = \arg\max_{\boldsymbol{\theta}_z} \sum_n \tilde{p}(\mathcal{D}_n|z) \log p_{\boldsymbol{\theta}_z}(\mathbf{a}_n|\mathbf{o}_n, z), = \arg\min_{\boldsymbol{\theta}_z} \sum_n \tilde{p}(\mathcal{D}_n|z) \|\boldsymbol{\mu}_{\boldsymbol{\theta}_z}(\mathbf{o}_n) - \mathbf{a}_n\|_2^2.$$

This optimization problem can be efficiently solved using gradient-based methods. Lastly, sampling from Gaussian distributions is well-known to be straightforward and efficient.

Parameterization. We experimented with two different parameterizations of the Gaussian expert means $\boldsymbol{\mu}_{\boldsymbol{\theta}_z}$, which we dub *single-head* and *multi-head*: For single-head, there is no parameter sharing between the different experts. Each expert has its own set of parameters $\boldsymbol{\theta}_z$. As a result, we learn N_z different multi-layer perceptrons (MLPs) $\boldsymbol{\mu}_{\boldsymbol{\theta}_z} : \mathbb{R}^{|\mathcal{O}|} \rightarrow \mathbb{R}^{|\mathcal{A}|}$, where N_z is the number of mixture components. In contrast, the multi-head parameterization uses a global set of parameters $\boldsymbol{\theta}$ for all experts and hence allows for feature sharing. We thus learn a single MLP $\boldsymbol{\mu}_{\boldsymbol{\theta}} : \mathbb{R}^{|\mathcal{O}|} \rightarrow \mathbb{R}^{N_z \times |\mathcal{A}|}$.

To compare both parameterizations, we conducted an ablation study where we evaluate the MoE policy on obstacle avoidance, table tennis and Franka kitchen. In order to have a similar number of parameters, we used smaller MLPs for single-head, that is, $1-4$ layers whereas for multi-head we used a 6 layer MLP. The results are shown in Table 9 and are generated using 30 components for the obstacle avoidance and table tennis task. The remaining hyperparameters are equal to the ones listed in the main manuscript. For Franka kitchen, we report the cumulative success rate and entropy for a different number of completed tasks. We report the mean and standard deviation calculated across 10 different seeds. Our findings indicate that, in the majority of experiments, the single-head parameterization outperforms the mutli-head alternative. Notably, we observed a substantial performance disparity, especially in the case of Franka kitchen.

Table 9: **Expert Parameterization Ablation**: We compare IMC with single- and multi-head expert parameterization. For further details, please refer to the accompanying text.

	OBSTACLE AVOIDANCE		TABLE TENNIS		FRANKA KITCHEN	
ARCHITECTURE	SUCCESS RATE (↑)	ENTROPY (↑)	SUCCESS RATE (↑)	DISTANCE ERR. (↓)	SUCCESS RATE (↑)	ENTROPY (↑)
SINGLE-HEAD	0.899 ± 0.035	0.887 ± 0.043	0.812 ± 0.039	0.168 ± 0.007	3.644 ± 0.230	6.189 ± 1.135
MULTI-HEAD	0.855 ± 0.053	0.930 ± 0.031	0.870 ± 0.017	0.153 ± 0.007	3.248 ± 0.062	4.657 ± 0.312

Expert Complexity. We conducted an ablation study to evaluate the effect of expert complexity on the performance of the IMC algorithm. The study involved varying the number of hidden layers in the single-head expert architecture while assessing the IMC algorithm's performance on the Franka kitchen task using the cumulative success rate and entropy. The results, presented in Figure 8, were obtained using the hyperparameters specified in the main manuscript. Mean and standard deviation were calculated across 5 different seeds. Our findings demonstrate a positive correlation between expert complexity and achieved performance.

E.2 Curriculum Pacing Sensitivity

To examine the algorithm's sensitivity to the curriculum pacing parameter η, we conducted an ablation study. Figure 9 presents the results obtained using 30 components for the obstacle avoidance and

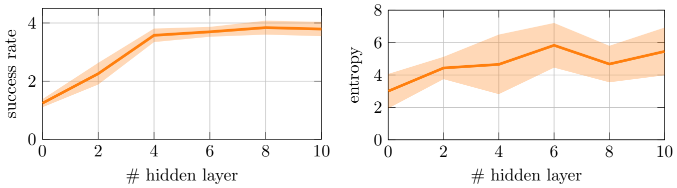
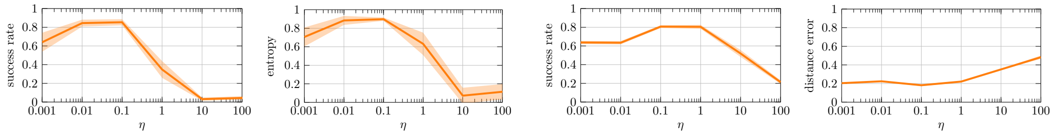


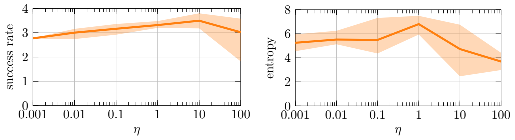
Figure 8: **Expert Complexity Ablation:** Evaluation of the IMC algorithm on the Franka kitchen task with varying numbers of hidden layers in the single-head expert architecture.

table tennis tasks, while maintaining the remaining hyperparameters as listed in the main manuscript. For the Franka kitchen task, we analyzed the cumulative success rate and entropy across varying numbers of completed tasks. The mean and standard deviation were calculated across 5 different seeds. Our findings reveal that the optimal value for η is dependent on the specific task. Nevertheless, the algorithm exhibits stable performance even when η values differ by an order of magnitude.



(a) Obstacle avoidance

(b) Table tennis



(c) Franka kitchen

Figure 9: **Curriculum Pacing Sensitivity:** Sensitivity analysis of the IMC algorithm for performance metrics in the obstacle avoidance, table tennis, and Franka kitchen tasks, considering varying curriculum pacing (η) values. The results illustrate the mean and standard deviation across 5 different seeds.

E.3 Inference Details

We provide pseudocode to further clarify the inference procedure of our proposed method (see Algorithm 2).

Algorithm 2 IMC Action Generation

```
Require: Curriculum weights $\{\tilde{p}(\mathcal{D}_n|z) \mid n \in \{1, ..., N\}, z \in \{1, ..., N_z\}\}$
Require: Expert parameter $\{\boldsymbol{\theta}_z \mid z \in \{1, ..., N_z\}\}$
Require: New observation $\mathbf{o}^*$
if not parameter_updated then
    $\boldsymbol{\phi}^* \leftarrow \arg\max_{\boldsymbol{\phi}} \sum_n \sum_z \tilde{p}(\mathcal{D}_n|z) \log g_{\boldsymbol{\phi}}(z|\mathbf{o}_n)$
    parameter_updated ← True
end if
Sample $z' \sim g_{\boldsymbol{\phi}^*}(z|\mathbf{o}^*)$
Sample $\mathbf{a}' \sim p_{\boldsymbol{\theta}_z}(\mathbf{a}|\mathbf{o}^*, z')$
Return $\mathbf{a}'$
```