Information Maximizing Curriculum: A Curriculum-Based Approach for Training Mixtures of Experts

Denis Blessing [1] **Onur Celik** [1] **Xiaogang Jia** [1] **Moritz Reuss** [1] **Maximilian Xiling Li** [1] **Rudolf Lioutikov** [1] **Gerhard Neumann** [1]

Abstract

Mixtures of Experts (MoE) are known for their ability to learn complex conditional distributions with multiple modes. However, despite their potential, these models are challenging to train and often tend to produce poor performance, explaining their limited popularity. Our hypothesis is that this under-performance is a result of the commonly utilized maximum likelihood (ML) optimization, which leads to mode averaging and a higher likelihood of getting stuck in local maxima. We propose a novel curriculum-based approach to learning mixture models in which each component of the MoE is able to select its own subset of the training data for learning. This approach allows for independent optimization of each component, resulting in a more modular architecture that enables the addition and deletion of components on the fly, leading to an optimization less susceptible to local optima. The curricula can ignore data-points from modes not represented by the MoE, reducing the mode-averaging problem. To achieve a good data coverage, we couple the optimization of the curricula with a joint entropy objective and optimize a lower bound of this objective. We evaluate our curriculum-based approach on a variety of multimodal behavior learning tasks and demonstrate its superiority over competing methods for learning MoE models and conditional generative models.

1. Introduction

Mixtures of experts (MoEs) are powerful models, that leverage a divide-and-conquer approach to conditional density estimation by assigning experts to smaller sub-tasks. They are capable of representing highly complex multimodal distributions but are inherently hard to train which often yields poor performance due to a sub-optimal optimization outcome. We hypothesize that these problems are due to training by maximizing the likelihood via gradient ascent (Bishop, 1994) or expectation maximization (Dempster et al., 1977). It is well known that maximum likelihood estimation corresponds to a moment projection which causes the model to average over modes that it cannot represent, leading to poor generative capabilities (Murphy, 2012). Moreover, these methods are often susceptible to poor solutions found due to local maxima, and finding an appropriate model complexity is difficult as the number of experts has to be specified a-priori.

In this work, we propose *Information Maximizing Curriculum* (IMC), a novel approach for training mixtures of experts that combines the information projection (Murphy, 2012) with curriculum learning (CL) to address the aforementioned problems with existing optimizing schemes. The information projection minimizes the reverse KL divergence which forces the model to ignore non-representable modes, leading to good generative models that are able to produce high quality samples. IMC assigns weights to samples according to their difficulty, resulting in reduced outlier sensitivity and better generalization capabilities (Bengio et al., 2009).

IMC employs a curriculum for each expert, which adapts to their performance and allows them to specialize on samples that they are able to represent. Moreover, the information projection is employed to compute the joint curriculum of all experts, which results in components that specialize to different subsets of the data. The curriculum also enables a modular architecture capable of online adaptation of the model complexity by adding experts to the model.

We show that our method is able to outperform state-of-the-art generative models on challenging multimodal conditional density estimation problems. In particular, we focus on complex behavior learning tasks where data is collected by human demonstrators. The inherent versatility in human behavior leads to highly multimodal data distributions. In our experiments, we assess the ability of the models to *i)* avoid mode averaging and *ii)* extract all modes present in the data distribution.

[1]Karlsruhe Institute of Technology, Germany. Correspondence to: Denis Blessing <denis.blessing@kit.edu>.

2. Preliminaries

Our approach heavily builds on mixtures of experts as well as minimizing Kullback-Leibler divergences. Hence, we will briefly review both concepts.

2.1. Mixtures of Experts

Mixtures of experts are conditional discrete latent variable models. Given some input $\mathbf{x} \in \mathcal{X}$ and target $\mathbf{y} \in \mathcal{Y}$, the marginal likelihood is decomposed into individual components o, that is,

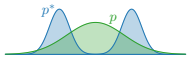
$$p(\mathbf{y}|\mathbf{x}) = \sum_o p(o|\mathbf{x})p(\mathbf{y}|\mathbf{x}, o).$$

The gating $p(o|\mathbf{x})$ is responsible for soft-partitioning the input space $\mathcal{X}$ into sub-regions where the corresponding experts $p(\mathbf{y}|\mathbf{x}, o)$ approximate the target density. Typically the experts and the gating are parameterized and learned by maximizing the likelihood via expectation-maximization (Dempster et al., 1977) or gradient ascent (Bishop, 1994). In order to sample from the marginal likelihood, that is, $\mathbf{y}' \sim p(\mathbf{y}|\mathbf{x}')$ for some $\mathbf{x}'$, we first sample a component index from the gating, i.e., $o' \sim p(o|\mathbf{x})'$. The component index selects the respective expert to obtain $\mathbf{y}' \sim p(\mathbf{y}|\mathbf{x}', o')$.

2.2. Moment and Information Projection

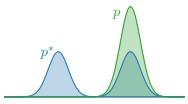
The Kullback-Leibler (KL) divergence (Kullback & Leibler, 1951) is a similarity measure for probability distributions and is defined as $D_{\mathrm{KL}}(p\|p') = \sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})/p'(\mathbf{x})$. Due to its asymmetry, the KL divergence offers two different optimization problems for fitting a model distribution p to a target distribution p^* (Murphy, 2012), that is,

$$\underbrace{\arg\min_p \; D_{\mathrm{KL}}(p^*\|p)}_{\text{M(oment)-Projection}}$$



and

$$\underbrace{\arg\min_p \; D_{\mathrm{KL}}(p\|p^*)}_{\text{I(nformation)-Projection}}.$$



The M-projection - or equivalently maximum likelihood estimation (MLE) (Bishop & Nasrabadi, 2006) - is *probability forcing*, meaning that the model is optimized to match the moments of the target distribution, causing it to average over modes that it cannot represent. In contrast, the I-projection is *zero forcing* which leads the model to ignore modes of the target distribution that it is not able to represent. The I-projection can be rewritten as maximization problem, i.e.,

$$\arg\max_p \; \mathbb{E}_{p(\mathbf{x})}[\log p^*(\mathbf{x})] + \mathcal{H}(\mathbf{x}). \tag{1}$$

Using this formulation, it can be seen that the optimization balances between fitting the target distribution and keeping the entropy $\mathcal{H}(\mathbf{x}) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$ high.

3. Information Maximizing Curriculum

In this section, we propose Information Maximizing Curriculum (IMC), a novel algorithm for training mixtures of experts. We motivate our optimization objective using a single expert model. Next, we generalize the objective to support an arbitrary number of experts. Thereafter, we discuss the optimization scheme and algorithmic details. Lastly, we explain how the model is used at inference time.

3.1. Objective for a Single Self-Paced Expert

We propose an objective that jointly learns a curriculum $p(\mathcal{D})$ and a parameterized expert distribution $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ with parameters $\boldsymbol{\theta}$. The curriculum is a categorical distribution $p(\mathcal{D}_n)$ over samples of a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, assigning probability mass to samples according to the performance of the expert. To allow the curriculum to ignore samples that the expert cannot represent, we build on the I-projection (see Equation 1). We therefore formulate the objective as

$$\max_{p(\mathcal{D}), \boldsymbol{\theta}} \; \mathbb{E}_{p(\mathcal{D})}[\log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})] + \eta\mathcal{H}(\mathcal{D}), \tag{2}$$

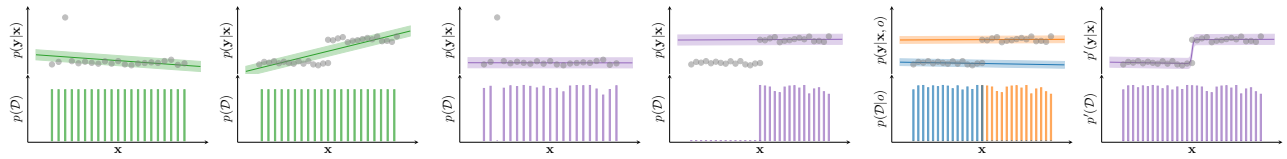
which is optimized for $p(\mathcal{D})$ and $\boldsymbol{\theta}$ in an alternating fashion. We additionally introduced a trade-off factor η that determines the pacing of the curriculum. For $\eta \to \infty$ the curriculum becomes uniform, exposing all samples to the expert and hence reducing to maximum likelihood estimation for $\boldsymbol{\theta}$. In contrast, if $\eta \to 0$ the curriculum concentrates on samples where the expert log-likelihood $\log p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ is highest. The objective can be solved in closed form for $p(\mathcal{D})$, i.e.,

$$p^*(\mathcal{D}_n) \propto p_{\boldsymbol{\theta}}(\mathbf{y}_n|\mathbf{x}_n)^{1/\eta}.$$

Maximizing the objective w.r.t $\boldsymbol{\theta}$ reduces to a weighted maximum-likelihood estimation for $\boldsymbol{\theta}$, that is,

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_n p(\mathcal{D}_n) \log p_{\boldsymbol{\theta}}(\mathbf{y}_n|\mathbf{x}_n).$$

The expert thus specializes to samples selected by the curriculum which in turn selects samples that lie within the representational capacity of the expert. As a result, the expert is paced by its own performance, thus the name *self-paced*. The optimization is repeated until the curriculum converges, meaning that the representational capacity of the expert is exhausted. The curriculum allows experts to ignore samples that they cannot represent which makes the model less sensitive to noise, outliers, and averaging over parts of the target function with high complexity, such as discontinuities. These properties are illustrated in Figure 1a



(a) (b) (c)

Figure 1. **Uniform vs. learned curriculum:** For a uniform curriculum $p(\mathcal{D})$, the expert optimization reduces to maximum likelihood estimation rendering $p(\mathbf{y}|\mathbf{x})$ prone to outliers and averaging effects (Figure (a)). Learning a curriculum counteracts these problems but ignores samples that the expert is not able to represent (Figure (b)). Introducing multiple curricula $p(\mathcal{D}|o)$ allows corresponding experts $p(\mathbf{y}|\mathbf{x}, o)$ to specialize to different subsets of the data and hence increasing the overall sample coverage of the joint curriculum $p'(\mathcal{D})$ and marginal likelihood $p'(\mathbf{y}|\mathbf{x})$ (Figure (c)).

and 1b. However, the downside of a self-pacing expert is the arbitrarily poor performance at samples that are ignored. This problem is alleviated by introducing multiple experts that specialize to different subsets of the data.

3.2. Objective for a Mixture of Self-Paced Experts

Assuming limited complexity, a single expert is likely to ignore a large amount of samples due to the zero-forcing property of the I-projection. Using multiple curricula and experts that specialize to different subsets of the data is hence a natural extension to the single expert model. To that end, we make two major modifications to Equation 2: Firstly, we use a mixture model with multiple components o where each component has its own curriculum, i.e., $p(\mathcal{D}) = \sum_o p(o)p(\mathcal{D}|o)$. Secondly, we employ an expert per component $p_{\boldsymbol{\theta}_o}(\mathbf{y}|\mathbf{x}, o)$, that is paced by the corresponding curriculum. The resulting objective function is given by

$$J(\boldsymbol{\psi}) = \mathbb{E}_{p(o)}\mathbb{E}_{p(\mathcal{D}|o)}[\log p_{\boldsymbol{\theta}_o}(\mathbf{y}|\mathbf{x}, o)] + \eta\mathcal{H}(\mathcal{D}), \quad (3)$$

where $\boldsymbol{\psi}$ summarizes the dependence on $p(o)$, $\{p(\mathcal{D}|o)\}_o$ and $\{\boldsymbol{\theta}_o\}_o$. However, Equation 3 is difficult to optimize as the entropy of the mixture model prevents us from updating each the curriculum of each component independently. Similar to (Arenz et al., 2018), we introduce an auxiliary distribution $q(o|\mathcal{D})$ to decompose the objective function into a lower bound $L(\boldsymbol{\psi}, q)$ and an expected D_{KL} term, that is,

$$J(\boldsymbol{\psi}) = L(\boldsymbol{\psi}, q) + \eta\mathbb{E}_{p(\mathcal{D})}D_{\mathrm{KL}}(p(o|\mathcal{D})\|q(o|\mathbf{x})), \quad (4)$$

with $p(o|\mathcal{D}) = p(\mathcal{D}|o)p(o)/p(\mathcal{D})$ and

$$L(\boldsymbol{\psi}, q) = \mathbb{E}_{p(o)}\big[\underbrace{\mathbb{E}_{p(\mathcal{D}|o)}[R_o(\mathcal{D})] + \eta\mathcal{H}(\mathcal{D}|o)}_{J_o(p(\mathcal{D}|o),\boldsymbol{\theta}_o)}\big] + \eta\mathcal{H}(o),$$

with $R_o(\mathcal{D}) = \log p_{\boldsymbol{\theta}_o}(\mathbf{y}|\mathbf{x}, o) + \eta \log q(o|\mathcal{D})$, allowing for independent updates for $p(\mathcal{D}|o)$ and $\boldsymbol{\theta}_o$ by maximizing the per-component objective function $J_o(p(\mathcal{D}|o), \boldsymbol{\theta}_o)$. Similar to the expectation-maximization algorithm, we introduced $q(o|\mathcal{D})$ as auxiliary distribution. Note that the lower bound decomposition holds for any distribution $q(o|\mathcal{D})$. A derivation can be found in Appendix B.1. Since $\mathbb{E}_{p(\mathcal{D})}D_{\mathrm{KL}}(p(o|\mathcal{D})\|q(o|\mathbf{x})) \geq 0$, L is a lower bound on J for $\eta \geq 0$. Please note that the per-component objective function J_o is very similar to Equation 2, i.e., the different experts specialize to samples selected by their curriculum $p(\mathcal{D}|o)$. However, J_o additionally contains $\log q(o|\mathcal{D})$ which prevents different curricula from assigning probability mass to the same samples. This property is further illustrated in Figure 1c. We follow the optimization scheme of the EM algorithm, that is, we iteratively maximize (M-step) and tighten the lower bound (E-step).

3.3. Maximizing the Lower Bound (M-Step)

We maximize the lower bound $L(\boldsymbol{\psi}, q)$ with respect to the mixture weights $p(o)$, curricula $p(\mathcal{D}|o)$ and expert parameters $\boldsymbol{\theta}_o$. We find closed form solutions for both, $p(o)$ and $p(\mathcal{D}|o)$ given by

$$p^*(o) \propto \exp\big(\mathbb{E}_{p(\mathcal{D}|o)}[R_o(\mathcal{D})/\eta] + \mathcal{H}(\mathcal{D}|o)\big), \quad (5)$$

and

$$p^*(\mathcal{D}_n|o) \propto \tilde{p}(\mathcal{D}_n|o) = \exp\big(R_o(\mathcal{D}_n)/\eta\big),$$

where $\tilde{p}(\mathcal{D}_n|o)$ are the optimal unnormalized curricula $p^*(\mathcal{D}_n|o)$. However, due to the hierarchical structure of $L(\boldsymbol{\psi}, q)$ we implicitly optimize for $p(o)$ when updating the curricula. This result is frequently used throughout this work and is formalized in Proposition 3.1. A proof is found in Appendix A.1.

Proposition 3.1. *Let $p^*(o)$ and $\tilde{p}(\mathcal{D}|o)$ be the optimal mixture weights and unnormalized curricula for maximizing $L(\boldsymbol{\psi}, q)$. It holds that*

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

The implicit updates of the mixture weights render the computation of $p^*(o)$ obsolete, reducing the optimization to computing the optimal (unnormalized) curricula $\tilde{p}(\mathcal{D}|o)$ and expert parameters $\boldsymbol{\theta}_o^*$. Maximizing the lower bound with

respect to the expert parameters results in a weighted maximum likelihood estimation, i.e.,

$$\boldsymbol{\theta}_o^* = \arg\max_{\boldsymbol{\theta}_o} \sum_n p(\mathcal{D}_n|o) \log p_{\boldsymbol{\theta}_o}(\mathbf{y}_n|\mathbf{x}_n, o), \quad (6)$$

where the curricula $p(\mathcal{D}_n|o)$ assign sample weights.

3.4. Tightening the Lower Bound (E-Step)

Tightening of the lower bound (also referred to as E-step) is done by minimizing the expected Kullback-Leibler divergence in Equation 4. Using the properties of the KL divergence, it can easily be seen that the lower bound is tight if for all $n \in \{1, ..., N\}$ $q(o|\mathbf{x}_n) = p(o|\mathcal{D}_n)$ holds. To obtain $p(o|\mathcal{D}_n)$ we leverage Bayes' rule, that is, $p(o|\mathcal{D}_n) = p^*(o)p^*(\mathcal{D}_n|o)/\sum_o p^*(o)p^*(\mathcal{D}_n|o)$. Using Proposition 3.1 we find that

$$p(o|\mathcal{D}_n) = \tilde{p}(\mathcal{D}_n|o)/\sum_o \tilde{p}(\mathcal{D}_n|o).$$

Please note that the lower bound is tight after every E-step as the KL divergence is set to zero. Thus, increasing the lower bound L maximizes the original objective J assuming that updates of $\boldsymbol{\theta}_o$ are not decreasing the expert log-likelihood $\log p_{\boldsymbol{\theta}_o}(\mathbf{y}|\mathbf{x}, o)$.

3.5. Automatic Per-Component Curriculum Pacing

Choosing a fixed curriculum pacing value η for applications where entropy and expert log-likelihood values change heavily during training can be a limiting assumption and might lead to sub-optimal results. Moreover, pacing all curricula with the same η can result in curricula that cover large subsets of the data while others degenerate, i.e., only cover few samples. In order to obtain an adaptive per-component curriculum pacing and alleviate problems with degrading curricula, we enforce a lower bound $\mathcal{H}_{\text{min}}$ on the entropy of the individual curricula. To that end, we frame the per-component objective J_o as constraint optimization problem, giving

$$\arg\max_{p(\mathcal{D}|o)} \mathbb{E}_{p(\mathcal{D}|o)}[R_o(\mathcal{D})] + \eta\mathcal{H}(\mathcal{D}|o), \text{ s.t. } \mathcal{H}(\mathcal{D}|o) \geq \mathcal{H}_{\text{min}}.$$

Using Lagrange duality (Boyd et al., 2004) we obtain a closed form solution by optimizing the Lagrangian function given by

$$p_{\xi_o}^*(\mathcal{D}|o) \propto \exp\Big(\frac{R_o(\mathcal{D})}{\xi_o + \eta}\Big),$$

with per-component Lagrangian multiplier ξ_o. The optimal value ξ_o^* is obtained by minimizing the Lagrangian dual function $g(\xi_o)$, that is, $\xi_o^* = \arg\min_{\xi_o > 0} g(\xi_o)$ with

$$g(\xi_o) = \xi_o\Big(\log \sum_n \exp\Big(\frac{R_o(\mathcal{D}_n)}{\xi_o + \eta}\Big) - \mathcal{H}_{\text{min}}\Big),$$

and thus $p^*(\mathcal{D}|o) = p_{\xi_o^*}^*(\mathcal{D}|o)$. Please note that ξ_o^* is obtained using a convex numerical optimizer. The minimum per-component entropy $\mathcal{H}_{\text{min}}$ is intuitive to choose as it translates into the number of samples N_s that a single curriculum should cover (uniformly) by setting $\mathcal{H}_{\text{min}} = \log N_s$.

3.6. Adding Components and Algorithmic Details

In contrast to mixtures of experts trained by either EM or backpropagation, we adapt the model complexity online by adding new components during the training procedure. Using such an online scheme for maximum likelihood based approaches is difficult as existing components are forced to cover all samples, making it challenging to initialize new components that improve the overall model performance.

We initialize the model with a single component and gradually increase the model complexity by adding more components to the mixture of experts model. Moreover, we use the convergence of the lower bound $L(\boldsymbol{\psi}, q)$ as criterion for adding new components to the model. After each M-step, the lower bound is evaluated using Corollary 3.1.1. See Appendix A.2 for a proof.

Corollary 3.1.1. *Consider the setup used in Proposition 3.1. For $p^*(o) \in \boldsymbol{\psi}$ and $\{p^*(\mathcal{D}|o)\}_o \in \boldsymbol{\psi}$ it holds that*

$$L\big(\boldsymbol{\psi}, q\big) = \eta \log \sum_o \sum_n \tilde{p}(\mathcal{D}_n|o).$$

Hence, if the difference between two subsequent iterations (i) and $(i-1)$ falls below a threshold ϵ, that is,

$$|\Delta L| = |L^{(i)}(\boldsymbol{\psi}, q) - L^{(i-1)}(\boldsymbol{\psi}, q)| \leq \epsilon,$$

a new component o_{new} is added. The full training procedure is outlined in Algorithm 1.

3.7. Inference

In order to perform inference, i.e., sampling from the model or computing expectations, we need to access the gating distribution for arbitrary inputs $\mathbf{x}$ which is not possible as $p(o|\mathcal{D})$ is only defined for samples contained in $\mathcal{D}$. We therefore leverage Corollary 3.1.2 to learn an inference network $g_{\boldsymbol{\phi}}(o|\mathbf{x})$ with parameters $\boldsymbol{\phi}$ by minimizing the KL divergence between $p(o|\mathcal{D})$ and $g_{\boldsymbol{\phi}}(o|\mathbf{x})$ under the joint curriculum $p(\mathcal{D})$ (see Appendix A.3 for a proof).

Corollary 3.1.2. *Consider the setup used in Proposition 3.1. For $p(o|\mathcal{D}) \propto p^*(o)p^*(\mathcal{D}|o)$ and $p(\mathcal{D}) = \sum_o p^*(o)p^*(\mathcal{D}|o)$ it holds that*

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{p(\mathcal{D})} D_{\text{KL}}\big(p(o|\mathcal{D}) \| g_{\boldsymbol{\phi}}(o|\mathbf{x})\big)$$
$$= \max_{\boldsymbol{\phi}} \sum_n \sum_o \tilde{p}(\mathcal{D}_n|o) \log g_{\boldsymbol{\phi}}(o|\mathbf{x}_n).$$

Algorithm 1 IMC training procedure

Require: Data $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$
Require: Curriculum pacing η
Require: Minimal entropy $\mathcal{H}_{\text{min}}$
Require: Max. number of components $N_{o,\text{max}}$
$N_o \leftarrow 1$
while $N_o \leq N_{o,\text{max}}$ **do**
  $q(o|\mathcal{D}_n) \leftarrow \tilde{p}(\mathcal{D}_n|o) / \sum_o \tilde{p}(\mathcal{D}_n|o)\ \forall n$
  for $o \leftarrow 1, .., N_o$ **do**
    $\tilde{p}_{\xi_o}(\mathcal{D}_n|o) \leftarrow \exp\left(\frac{R_o(\mathcal{D}_n)}{\xi_o + \eta}\right)\ \forall n$
    $\xi_o^* \leftarrow \arg\min_{\xi_o > 0}\ g(\xi_o)$
    $\boldsymbol{\theta}_o^* \leftarrow \arg\max_{\boldsymbol{\theta}_o} \sum_n p^*(\mathcal{D}_n|o) \log p_{\boldsymbol{\theta}_o}(\mathbf{y}_n|\mathbf{x}_n, o)$
  end for
  if $|\Delta L| \leq \epsilon$ **then**
    add_component()
    $N_o \leftarrow N_o + 1$
  end if
end while

Training a gating network can be cumbersome for applications where we often switch between training and inference (e.g. due to new data acquisition), since adding components requires learning a new gating network from scratch due to varying network sizes. We counteract this problem by using an output dimension equal to the maximal number of components $N_{o,\text{max}}$ in the MoE model. Using a masking layer, i.e., a binary vector that sets the current number of components N_o to 1 and 0 otherwise allows to preserve the learned representation for existing components when adding a new component.

4. Related Work

Mixtures of Experts. The mixture of experts model was first proposed by Jacobs et al. (1991) and used expectation maximization (Dempster et al., 1977) for optimizing the model parameters. Several studies are dedicated to increasing the flexibility of the model (Jordan & Jacobs, 1994; Waterhouse, 1998; Bishop & Svensén, 2012). On another note, Bishop (1994) introduced the mixture density network (MDN) which uses a neural network whose parameters are shared between gating and experts, allowing for an end-to-end training using the backpropagation algorithm (Rumelhart et al., 1986). All of these works maximize the likelihood to optimize the model parameters which corresponds to a moment projection. This differs from our approach which is inspired by the information projection. Recently, Becker et al. (2020) introduced expected information maximization (EIM), an approach for computing the expected information projection based on samples from a dataset. While EIM was mainly introduced in the context of density estimation, the authors mention that the algorithm is also applicable to conditional models. However, EIM relies on an intermediate density ratio estimation step, causing stability issues and preventing scaling to high dimensional problems. We, therefore, do not consider EIM as a competitive baseline. For an elaborate survey on mixture of experts models, the reader is referred to Yuksel et al. (2012) and Masoudnia & Ebrahimpour (2014).

Curriculum Learning. Bengio et al. (2009) introduced curriculum learning (CL) as a new paradigm for training machine learning models by gradually increasing the difficulty of samples that are exposed to the model. Several studies followed this definition (Spitkovsky et al., 2009; Soviany et al., 2022; Chen & Gupta, 2015; Tudor Ionescu et al., 2016; Pentina et al., 2015; Shi et al., 2015; Zaremba & Sutskever, 2014). Other studies used the term curriculum learning for gradually increasing the model complexity (Karras et al., 2017; Morerio et al., 2017; Sinha et al., 2020) or task complexity (Caubrière et al., 2019; Florensa et al., 2017; Lotter et al., 2017; Sarafianos et al., 2017). All of these approaches assume that the difficulty-ranking of the samples is known a-priori. In contrast, we consider dynamically adapting the curriculum according to the learning progress of the model which is known as self-paced learning (SPL). Pioneering work in SPL was done by Kumar et al. (2010) which is related to our work in that the authors propose to update the curriculum as well as model parameters iteratively. However, their method is based on maximum likelihood which is different from our approach. Moreover, their algorithm is restricted to latent structural support vector machines. For a comprehensive survey on curriculum learning, the reader is referred to (Soviany et al., 2022).

5. Experiments

We evaluate IMC on challenging conditional density estimation tasks. To that end, we focus on behavior learning tasks with human-collected data. The inherent versatility in human behavior introduces high variability and outliers to data, inducing complex multimodal distributions. Hence, our experiments assess the ability of models to *i)* avoid mode averaging and *ii)* cover all modes present in the data distribution. For all experiments, we employ conditional Gaussian experts, i.e., $p_{\boldsymbol{\theta}_o}(\mathbf{y}|\mathbf{x}, o) = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_{\boldsymbol{\theta}_o}(\mathbf{x}), \sigma^2\mathbf{I})$ due to efficient sample routines and the simplicity of the resulting optimization problem. Please note that we parameterize the expert means $\boldsymbol{\mu}_{\boldsymbol{\theta}_o}$ as (shallow) neural networks. Moreover, we use a fixed variance of $\sigma^2 = 1$ for all experiments as full Gaussian likelihood optimization often results in unstable updates (Guo et al., 2017).

We compare our method to state-of-the-art generative models including denoising diffusion probabilistic models (DDPM) (Ho et al., 2020), normalizing flows (NF) (Papa-

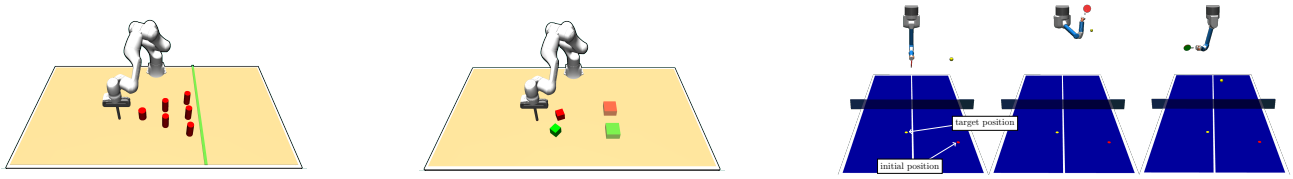


Figure 2. **Behavior learning environments:** Visualization of the obstacle avoidance task (left), the block pushing task (middle), and the table tennis task (right).

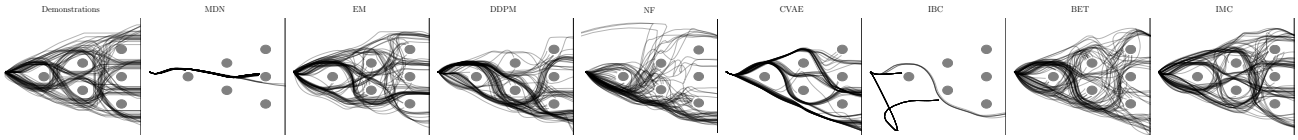


Figure 3. **Obstacle Avoidance:** Visualization of 100 end-effector trajectories for all trained models.

makarios et al., 2021) and conditional variational autoencoders (CVAE). Moreover, we consider energy-based models for behavior learning (IBC) (Florence et al., 2022) and the recently proposed behavior transformer (BeT) (Shafiullah et al., 2022). Lastly, we compare against mixture of experts trained using expectation maximization (EM) (Jacobs et al., 1991) and backpropagation (MDN) (Bishop, 1994). We extensively tune the hyperparameters of the baselines using Bayesian optimization (Snoek et al., 2012) on all experiments. We report the mean and the standard deviation over ten random seeds for all experiments. For a detailed explanation of tasks, data, performance metrics and hyperparameters see Appendix C.

5.1. Obstacle Avoidance

The obstacle avoidance environment is visualized in Figure 2 (left) and consists of a seven DoF Franka Emika Panda robot arm equipped with a cylindrical end effector simulated using the MuJoCo physics engine (Todorov et al., 2012). The task is to reach the green finish line without colliding with one of the six obstacles. The dataset contains four human demonstrations for all 24 ways of avoiding obstacles and completing the task which are collected using a game-pad controller and inverse kinematics (IK) in the xy-plane amounting to 7.3k $(\mathbf{x}, \mathbf{y})$ pairs. The inputs $\mathbf{x} \in \mathbb{R}^4$ contain the end-effector position and velocity of the robot. The targets $\mathbf{y} \in \mathbb{R}^2$ represent the desired position of the robot. To evaluate the susceptibility to mode averaging, we use the *success rate*, i.e., the percentage of trajectories that reach the finish line. Moreover, we assess a model's ability to learn multimodal distributions by computing the *entropy* of the categorical distribution that contains the probabilities of a model completing the different ways of avoiding obstacles. For more details see Appendix C.1.1. For a visual illustration of end-effector trajectories see Figure 3. The results are shown in Table 1 and are generated using 1000 evaluation trajectories for each seed. IMC achieves a superior success rate and entropy compared to the baselines. CVAE closely follows the success rate of IMC but lacks the ability to discover modes in the data distribution, indicated by the low entropy value. In contrast, EM achieves an entropy similar to IMC but is inferior with respect to the success rate.

5.2. Block Pushing

The block pushing environment is visualized in Figure 2 (middle) and uses the setup explained in Section 5.1 with the 2-D gamepad controller. However, the robot manipulator is tasked to push blocks into target zones. Having two blocks and target zones amounts to four different push sequences. See Figure 4 for an example. We consider 30 different block configurations $\boldsymbol{c}$ (i.e., initial orientation and position) that are uniformly sampled from a configuration space. Using a game-pad controller we recorded four trajectories for all push sequences and block configurations amounting to a total of $30 \times 4 \times 4 = 480$ demonstrations and thus 100k $(\mathbf{x}, \mathbf{y})$ pairs. The inputs $\mathbf{x} \in \mathbb{R}^{16}$ contain information about the robot's state and the block configurations. The targets $\mathbf{y} \in \mathbb{R}^2$ represent the desired position of the robot. We evaluate the models using three different metrics: First, the *success rate* which is the proportion of trajectories that manage to push both boxes to the target zones. Next, the expected *entropy* over a categorical distribution containing the probabilities of a model completing different push sequences conditioned on the initial block configuration $\mathbf{c}$. Lastly, to evaluate the performance on non-successful trajectories, we employ the *distance error*, that is, the distance from the blocks to the target zones at the end of a trajectory. The success rate and distance error indicate whether a model is able to avoid averaging over different behavior. Moreover, the entropy assesses the ability to represent multimodal data distributions by completing different push-sequences for the same configuration. See Appendix C.1.2 for more details. The results are reported in Table 1 and generated simulating 16 evaluation trajectories for all 30 contexts per seed. IMC

Table 1. **Result Table**: Performance comparison between various generative models on three behavior learning tasks.

	Obstacle Avoidance		Block Pushing			Table Tennis	
	success rate (↑)	Entropy (↑)	success rate (↑)	Entropy (↑)	Distance Error (↓)	success rate (↑)	Distance Error (↓)
MDN	0.200±0.421	0.000±0.000	0.000±0.000	0.000±0.000	0.360±0.005	0.031±0.013	0.549±0.056
EM	0.675±0.033	0.902±0.035	0.154±0.021	0.317±0.057	0.192±0.006	0.725±0.042	0.220±0.012
DDPM	0.719±0.075	0.638±0.079	0.075±0.016	0.053±0.035	0.207±0.013	0.866±0.010	0.185±0.007
NF	0.313±0.245	0.349±0.208	0.001±0.001	0.000±0.000	0.346±0.034	0.422±0.035	0.371±0.013
CVAE	0.853±0.113	0.465±0.183	0.041±0.014	0.019±0.018	0.224±0.010	0.620±0.050	0.320±0.010
IBC	0.379±0.411	0.098±0.131	0.000±0.000	0.000±0.000	0.357±0.041	0.567±0.030	0.310±0.010
BET	0.504±0.076	0.837±0.066	0.329±0.047	**0.595±0.077**	0.163±0.013	0.758±0.025	0.235±0.011
IMC	**0.855±0.053**	**0.930±0.031**	**0.413±0.060**	0.441±0.101	**0.158±0.018**	**0.870±0.017**	**0.153±0.007**

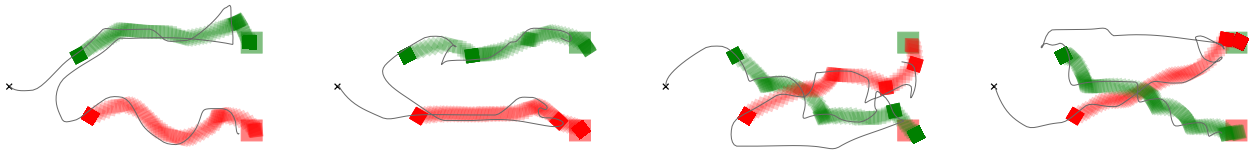


Figure 4. **Block pushing:** Top view of four different push sequences for the same initial block configuration $\mathbf{c}$. Following the gray line from the black cross visualizes the end-effector trajectory of the robot manipulator. The small rectangles indicate different box configurations in the push sequence while the big rectangles mark the target zones.

achieves superior success rate and distance error while BET has the highest entropy. The difficulty of the task is reflected by the low success rates of most models. Besides being a challenging manipulation task, the high task complexity is caused by having various sources of multimodality in the data distribution: First, the inherent versatility in human behavior. Second, multiple human demonstrators, and lastly different push sequences for the same block configuration.

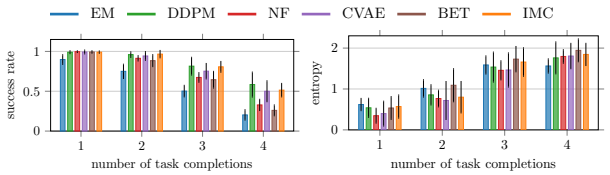


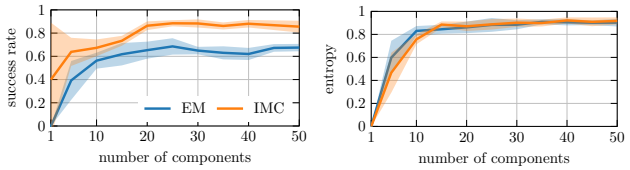
Figure 5. **Franka Kitchen:** Performance comparison between various generative models.

5.3. Franka Kitchen

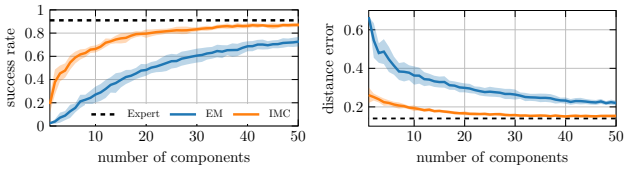
The Franka kitchen environment was introduced by Gupta et al. (2019) and uses a seven DoF Franka Emika Panda robot with a two DoF gripper to interact with a simulated kitchen environment. The corresponding dataset contains 566 human-collected trajectories collected using a virtual reality setup amounting to 128k $(\mathbf{x}, \mathbf{y})$ pairs. Each trajectory executes a sequence completing four out of seven different tasks. The inputs $\mathbf{x} \in \mathbb{R}^{30}$ contain information about position and orientation of the task-relevant objects in the environment. The targets $\mathbf{y} \in \mathbb{R}^9$ represent the control signals for the robot and gripper. To asses a model's ability to avoid mode averaging we again use the *success rate* over the number of tasks solved within one trajectory. Completing tasks in different orders introduces multimodality into the data distribution which is assessed by the *entropy* of the distribution over completed task sequences. For more details see Appendix C.1.3. The results are shown in Figure 5 and are generated using 100 evaluation trajectories for each seed. There are no results reported for IBC and MDN as we did not manage to obtain reasonable results. All models except for EM manage to complete one and two tasks with a success rate close to 1. For three and four task completions DDPM has the highest success rate closely followed by IMC and CVAE. EM has the highest entropy for one task. For two, three and four tasks BET achieves superior entropy, being slightly ahead of IMC and DDPM.

5.4. Table Tennis

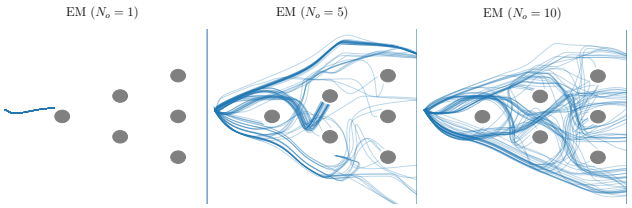
The table tennis environment is visualized in Figure 2 (right) and consists of a seven DOF robot arm equipped with a table tennis racket and is simulated using the MuJoCo physics engine. The goal is to return the ball to varying target positions after it is launched from a randomized initial position. Although not collected by human experts, the 5000 demonstrations are generated using a reinforcement learning agent that is optimized for highly multimodal behavior such as backhand and forehand strokes (Celik et al., 2022). Each demonstration consists of an input $\mathbf{x} \in \mathbb{R}^4$ defining the initial and target ball position. Movement primitives (MPs) (Paraschos et al., 2013) are used to describe the joint space



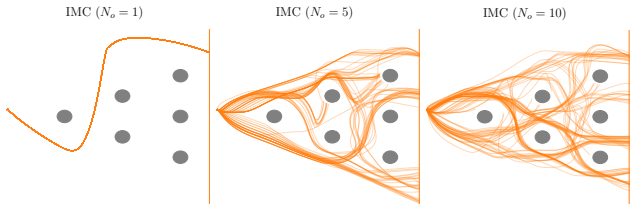
(a) Obstacle avoidance performance comparison.



(b) Table tennis performance comparison.



(c) Obstacle avoidance end-effector trajectories for EM.



(d) Obstacle avoidance end-effector trajectories for IMC.

Figure 6. **Ablation study:** Performance comparison between the EM and IMC algorithm for an increasing number of components.

trajectories of the robot manipulator using two basis functions per joint and thus $\mathbf{y} \in \mathbb{R}^{14}$. We evaluate the model performance using the *success rate*, that is, how frequently the ball is returned to the other side. Moreover, we employ the *distance error*, i.e., the euclidean distance from the landing position of the ball to the target position. Both metrics reflect if a model is able to avoid averaging over different movements. For this experiment, there is no metric to assess multimodality as it is difficult to quantify the versatility in the model behavior. The results are shown in Table 1 and are generated using 500 different initial and target positions. Please note that the reinforcement learning agent used to generate the data achieves an average success rate of 0.91 and a distance error of 0.14. This performance is closely followed by IMC which achieves superior performance compared to the other methods.

5.5. Ablation Studies

Additionally, we compare the performance of IMC with EM for a varying number of components on the obstacle avoidance and table tennis task. Please note that the incremental component adding scheme of IMC allows for evaluating the model performance after additional components are added. In contrast, the model is trained from scratch when employing EM for evaluating the performance using a different number of components. The results are shown in Figure 6 and highlight the properties of the moment and information projection: Using limited model complexity, e.g. 1 or 5 components, EM suffers from mode averaging, resulting in poor performances (Figure 6a and Figure 6b). This is further illustrated in Figure 6c. In contrast, the zero forcing property of the information projection allows IMC to avoid mode averaging (see Figure 6d) which is reflected in the success rates and distance error for a small number of components. The performance gap between EM and IMC for high model complexities suggests that EM still suffers from averaging problems. Moreover, the results show that IMC needs fewer components to achieve the same performance as EM.

6. Conclusion

We introduced Information Maximizing Curriculum (IMC), a novel approach to learning mixture of experts models (MoE). IMC is a curriculum-based approach that allows each expert to select its own subset of the training data for learning. The curriculum allows the MoE model to automatically ignore data points that it can not represent, which reduces the susceptibility to local optima and in particular to mode-averaging, a common problem associated with maximum likelihood-based optimization for multimodal density estimation. The maximization of the entropy of the joint curriculum of all experts incentivizes the MoE to cover all data samples. IMC is able to adapt the model complexity online by adding more experts during training which is enabled by the proposed objective. We motivated our objective for a single expert and generalized it to the MoE case. We showed that our method is able outperform existing optimization schemes for MoE and state-of-the art generative models on challenging multimodal conditional density estimation problems. In particular, we employed behavior learning tasks to show that IMC is able *i)* avoid mode averaging and *ii)* extract all modes present in the data distribution.

Despite introducing a lower bound on the per-component entropy of the curricula, which helps to tune the curriculum pacing η, we find that it can still be difficult to determine an appropriate value. In future work, we plan to tackle this issue.

References

Arenz, O., Neumann, G., and Zhong, M. Efficient gradient-free variational inference using policy search. In *International conference on machine learning*, pp. 234–243. PMLR, 2018.

Becker, P., Arenz, O., and Neumann, G. Expected information maximization: Using the i-projection for mixture density estimation. *arXiv preprint arXiv:2001.08682*, 2020.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Bishop, C. M. Mixture density networks. 1994.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Bishop, C. M. and Svensén, M. Bayesian hierarchical mixtures of experts. *arXiv preprint arXiv:1212.2447*, 2012.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Caubrière, A., Tomashenko, N., Laurent, A., Morin, E., Camelin, N., and Estève, Y. Curriculum-based transfer learning for an effective end-to-end spoken language understanding and domain portability. *arXiv preprint arXiv:1906.07601*, 2019.

Celik, O., Zhou, D., Li, G., Becker, P., and Neumann, G. Specializing versatile skill libraries using local mixture of experts. In *Conference on Robot Learning*, pp. 1423–1433. PMLR, 2022.

Chen, X. and Gupta, A. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1431–1439, 2015.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, pp. 158–168. PMLR, 2022.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pp. 482–495. PMLR, 2017.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Jordan, M. I. and Jacobs, R. A. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2): 181–214, 1994.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Kullback, S. and Leibler, R. A. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86, 1951.

Kumar, M., Packer, B., and Koller, D. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

Lotter, W., Sorensen, G., and Cox, D. A multi-scale cnn and curriculum learning strategy for mammogram classification. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pp. 169–177. Springer, 2017.

Masoudnia, S. and Ebrahimpour, R. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2): 275–293, 2014.

Morerio, P., Cavazza, J., Volpi, R., Vidal, R., and Murino, V. Curriculum dropout. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3544–3552, 2017.

Murphy, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

Paraschos, A., Daniel, C., Peters, J. R., and Neumann, G. Probabilistic movement primitives. *Advances in neural information processing systems*, 26, 2013.

Pentina, A., Sharmanska, V., and Lampert, C. H. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5492–5500, 2015.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Sarafianos, N., Giannakopoulos, T., Nikou, C., and Kakadiaris, I. A. Curriculum learning for multi-task classification of visual attributes. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2608–2615, 2017.

Shafiullah, N. M. M., Cui, Z. J., Altanzaya, A., and Pinto, L. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2206.11251*, 2022.

Shi, Y., Larson, M., and Jonker, C. M. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154, 2015.

Sinha, S., Garg, A., and Larochelle, H. Curriculum by smoothing. *Advances in Neural Information Processing Systems*, 33:21653–21664, 2020.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N. Curriculum learning: A survey. *International Journal of Computer Vision*, pp. 1–40, 2022.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. Baby steps: How "less is more" in unsupervised dependency parsing. 2009.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Tudor Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D. P., and Ferrari, V. How hard can it be? estimating the difficulty of visual search in an image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2157–2166, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Waterhouse, S. R. *Classification and regression using mixtures of experts*. PhD thesis, Citeseer, 1998.

Yuksel, S. E., Wilson, J. N., and Gader, P. D. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

Zaremba, W. and Sutskever, I. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Zhou, Y., Gao, J., and Asfour, T. Movement primitive learning and generalization: Using mixture density networks. *IEEE Robotics & Automation Magazine*, 27(2):22–32, 2020.

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 3.1.1

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 3.1.2

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

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

which concludes the proof. □

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.

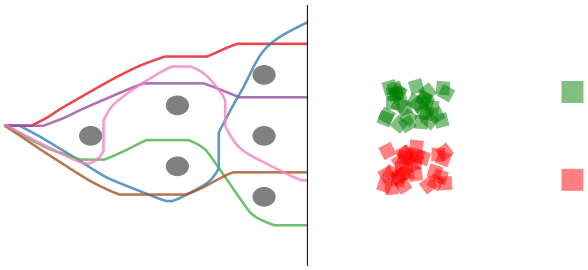


Figure 7. The left figure shows 6 out of 24 ways of completing the obstacle avoidance task. The right figure shows all 30 initial block configurations used for the block pushing task.

C. Experiment Setup

C.1. Environments and Datasets

C.1.1. Obstacle Avoidance

Dataset. The obstacle avoidance dataset contains 96 trajectories resulting in a total of 7.3k $(\mathbf{x}, \mathbf{y})$ pairs. The inputs $\mathbf{x} \in \mathbb{R}^4$ contain the end-effector position and velocity in Cartesian space. Please note that the height of the robot is fixed. The targets $\mathbf{y} \in \mathbb{R}^2$ 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 480 trajectories resulting in a total of 100k $(\mathbf{x}, \mathbf{y})$ pairs. The inputs $\mathbf{x} \in \mathbb{R}^{16}$ 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. Please note that the height of the robot is fixed. The targets $\mathbf{y} \in \mathbb{R}^2$ represent the desired position of the robot. For all 30 initial block configurations $\mathbf{c}$, i.e., position and orientation, we record four trajectories for all (four) push sequences. This task is similar to the one proposed in (Florence et al., 2022). 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 extract 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 by Gupta et al. (2019). It contains 566 human-collected trajectories resulting in a total of 128k $(\mathbf{x}, \mathbf{y})$ pairs. The inputs $\mathbf{x} \in \mathbb{R}^{30}$ contain information about position and orientation of the task-relevant objects in the environment. The targets $\mathbf{y} \in \mathbb{R}^9$ 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.

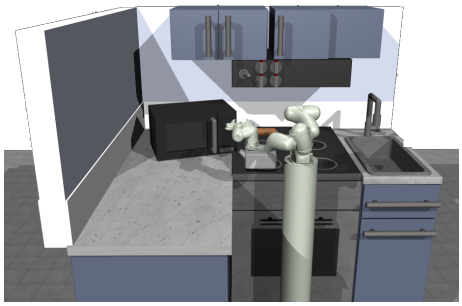


Figure 8. **Franka kitchen environment.**

C.1.4. TABLE TENNIS

Dataset. The table tennis dataset contains 5000 $(\mathbf{x}, \mathbf{y})$ pairs. The inputs $\mathbf{x} \in \mathbb{R}^4$ contain the coordinates of the initial and target ball position as projection on the table. Movement primitives (MPs) (Paraschos et al., 2013) are used to describe the joint space trajectories of the robot manipulator using two basis functions per joint and thus $\mathbf{y} \in \mathbb{R}^{14}$.

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.2. IMC Details and Hyperparameter

IMC employs a parameterized inference network and conditional Gaussian distributions to represent experts. For the latter, we also use a fixed variance of 1 and parameterize the means as shallow neural networks. In every M-step, we optimize the experts for 5 epochs. We add a new component after performing 5 E- and M-steps or if the lower bound $L(\boldsymbol{\psi}, q)$ converges. Moreover, we use automatic per-component curriculum pacing (Section 3.5) for all experiments. For the table tennis and obstacle avoidance task, we experimented with experts with 1 and 2 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 considered 3 and 4 layers. We found that 3 layers with 64 neurons yield the best results for the block pushing task and 4 layer with 100 for Franka kitchen. For all experiments except the kitchen task, we used an effective number of samples $N_s = 50$ to calculate the minimal per-component entropy $\mathcal{H}_{\text{min}}$. For the kitchen task, we used $N_s = 150$. We found that an expert learning rate of 10^{-3} leads to good results on all experiments. 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, $\eta \approx 45$ for block pushing and $\eta = 1$ for Franka kitchen and $\eta = 1$ for table tennis. To find an appropriate model complexity we added up to 50 components for the obstacle avoidance and table tennis task and 30 for the block pushing and Franka kitchen task. For the former two, the best results were obtained using 50 components. For the latter 10 components were sufficient to achieve the results reported in the main manuscript. We always evaluated the model after adding 5 components.

Table 2. **IMC Hyperparameter.**

PARAMETER	VALUE
EXPERT LEARNING RATE	10^{-3}
EXPERT BATCHSIZE	512
EXPERT VARIANCE ($\sigma^2_{\mathbf{y}}$)	1
INFERENCE NET HIDDEN LAYER	4
INFERENCE NET HIDDEN UNITS	200
INFERENCE NET EPOCHS	800
INFERENCE NET LEARNING RATE	10^{-3}
INFERENCE NET BATCHSIZE	512

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 (Jacobs et al., 1991) is identical to the one optimized with IMC: We employ a parameterized inference network and conditional Gaussian distributions to represent experts. For the latter, we also use a fixed variance of 1 and parameterize the means as shallow neural networks. For all experiments, we train the model for 100 EM steps or until the lower bound on the marginal likelihood converges. Equal to IMC, EM trains each expert for 5 epochs per M-step. However, the two approaches differ in the optimization scheme: While IMC trains the inference network once, EM updates the parameters in every M-step. We therefore optimize the inference network for 8 epochs resulting in a total of 800 epochs using 100 expectation maximization steps which is equivalent to the number used for IMC. Moreover, EM initialized all components at the beginning of the training whereas IMC incrementally adds components during training. For the table tennis and obstacle avoidance task we experimented using experts with 1 and 2 layer neural networks. We found that using 1 layer with 64 neurons performed best. For the block pushing and Franka kitchen experiments we

considered 3 and 4 layers. We found that 3 layers with 100 neurons yield the best results. To find an appropriate model complexity we tested up to 50 components for the obstacle avoidance and table tennis task and 30 for the block pushing and Franka kitchen task. For the former two, the best results were obtained using 25 and 50 components respectively. For the latter 20 and 25 components. For the remaining hyperparamter choices see Table 3.

Table 3. **EM Hyperparameter.**

PARAMETER	VALUE
EXPERT LEARNING RATE	10^{-3}
EXPERT BATCHSIZE	512
EXPERT VARIANCE (σ_y^2)	1
INFERENCE NET HIDDEN LAYER	4
INFERENCE NET HIDDEN UNITS	200
INFERENCE NET EPOCHS	800
INFERENCE NET LEARNING RATE	10^{-3}
INFERENCE NET BATCHSIZE	512

Mixture Density Network (MDN). The mixture density network (Bishop, 1994) 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 backbone 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 for the obstacle avoidance and table tennis task and 30 for the block pushing and Franka kitchen task. We found that the number of experts heads did not significantly influence the results, further indicating the 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 (Zhou et al., 2020) 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 4.

Denoising Diffusion Probabilistic Models (DDPM). We consider the denoising diffusion probabilistic model proposed by (Ho et al., 2020). Following common practice we parameterize the model as neural network with a sinusoidal positional encoding for the diffusion steps (Vaswani et al., 2017). Moreover, we use the a cosine-based variance scheduler proposed by (Nichol & Dhariwal, 2021). For further details on hyperparameter choices see Table 5.

Normalizing Flow (NF). For all experiments, we build the normalizing flow by stacking masked autoregressive flows (Papamakarios et al., 2017) paired with permutation layers (Papamakarios et al., 2021). As base distribution, we use a conditional isotropic Gaussian. Following common practice, we optimize the model parameters by maximizing its likelihood. See Table 6 for a list of hyperparamters.

Table 4. **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_y^2)	–	1
MAX. EPOCHS	–	2000
BATCHSIZE	–	512

Table 5. **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

Table 6. **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 by Sohn et al. (2015). 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 by Higgins et al. (2017).

Table 7. **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 by Florence et al. (2022) 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 (Florence et al., 2022). For sampling, we use gradient-based Langevin MCMC (Du & Mordatch, 2019). Despite our effort, we could not achieve good results with IBC. A list of hyperparameters is shown in Table 8.

Table 8. **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). Recently, Shafiullah et al. (2022) proposed the behavior transformer which employs a minGPT transformer (Brown et al., 2020) 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 9.

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.

Table 9. **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}

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

$$q(o|\mathbf{x}_n) = p(o|\mathbf{x}_n, \mathbf{y}_n) = \frac{p_{\boldsymbol{\theta}}(\mathbf{y}_n|\mathbf{x}_n, o)p(o|\mathbf{x}_n)}{\sum_o p_{\boldsymbol{\theta}_o}(\mathbf{y}_n|\mathbf{x}_n, o)p(o|\mathbf{x}_n)}, \tag{7}$$

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

$$q(o|\mathbf{x}_n)^{(i+1)} = \frac{p_{\boldsymbol{\theta}}(\mathbf{y}_n|\mathbf{x}_n, o)q(o|\mathbf{x}_n)^{(i)}}{\sum_o p_{\boldsymbol{\theta}}(\mathbf{y}_n|\mathbf{x}_n, o)q(o|\mathbf{x}_n)^{(i)}},$$

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

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

Consequently, we see that $q(o|\mathbf{x}) = q(o|\mathcal{D})$ 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.

$$\max_{\boldsymbol{\theta}_o} \sum_{n=1}^{N} q(o|\mathbf{x}_n) \log p_{\boldsymbol{\theta}_o}(\mathbf{y}_n|\mathbf{x}_n, o),$$

whereas IMC uses the curricula as weights, that is,

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

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

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

for the EM algorithm and

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

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 $\tilde{p}(\mathcal{D}_n|o)$ are zero for all components.