DISCRETE VARIATIONAL AUTOENCODING VIA POLICY SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete latent bottlenecks in variational autoencoders (VAEs) offer high bit efficiency and can be modeled with autoregressive discrete distributions, enabling parameter-efficient multimodal search with transformers. However, discrete random variables do not allow for exact differentiable parameterization; therefore, discrete VAEs typically rely on approximations, such as Gumbel-Softmax reparameterization or straight-through gradient estimates, or employ high-variance gradient-free methods such as REINFORCE that have had limited success on high-dimensional tasks such as image reconstruction. Inspired by popular techniques in policy search, we propose a training framework for discrete VAEs that leverages the natural gradient of a non-parametric encoder to update the parametric encoder without requiring reparameterization. Our method, combined with automatic step size adaptation and a transformer-based encoder, scales to challenging datasets such as ImageNet and outperforms both approximate reparameterization methods and quantization-based discrete autoencoders in reconstructing high-dimensional data from compact latent spaces, achieving a 20% improvement on FID Score for ImageNet 256.

1 INTRODUCTION

Discrete representations play a critical role in numerous fields, including telecommunications, biology, and robotics. While real-world data is often continuous, introducing a discrete latent bottleneck can offer unique advantages, such as increased bit-efficiency, multimodal modeling, and the ability to leverage tools from combinatorial optimization. These properties make discrete representations particularly useful for tasks that benefit from structured or compact latent spaces. However, encoding continuous information into discrete representations and decoding it back remains a challenging endeavor. In particular, gradient-based optimization requires a differentiable transformation to map continuous inputs to discrete latents, but this is impeded by the non-differentiability of operations like rounding or taking the argmax, which are essential for obtaining hard assignments. Because of this difficulty, we are often left with approximate techniques, such as Gumbel Softmax reparameterization and straight-through esimation Jang et al. (2016).

However, approximate reparameterization with Gumbel-Softmax Jang et al. (2016) is sensitive to its temperature hyperparameter, forcing a trade-off between increased gradient variance at low temperatures and large approximation error at high temperatures. Furthermore, for larger bottlenecks, backpropagating the soft-assignments through the autoregressive sampling process has high memory footprint and can suffer from vanishing gradients. Variance-reduction schemes such as GR-MCK Paulus et al. (2020) help but do not fully resolve the scaling problem. On the other hand, vector-quantization methods, such as VQ-VAE Van Den Oord et al. (2017) and FSQ Mentzer et al. (2023)) avoid the problems of discrete reparameterization, but still rely on approximations due to the use of straight-through estimation. Furthermore, their latent distribution is intractable, which prevents them from maximizing the latent entropy in the ELBO and demands special training objectives. Score-function-based gradient estimators, such as REBAR (Tucker et al., 2017) or Muprop (Gu et al., 2015), are a promising alternative as they can obtain unbiased gradients to optimize the ELBO in the discrete setting. However, due to their high gradient variance, the success of these methods has so far been limited.

To overcome these limitations, we propose Discrete Autoencoding via Policy Search (DAPS), a training framework that optimizes the ELBO for discrete, autoregressive encoders without any discrete reparameterization or straight-through tricks. DAPS casts encoder learning as a KL-regularized policy-search problem: we form a closed-form, nonparametric target distribution $q*$ and update a parametric encoder $q_{\boldsymbol{\theta}}$ by weighted maximum likelihood, avoiding backpropagation through the sampling path. A single scalar trust-region parameter η is adapted automatically using an effective sample size (ESS) objective, yielding stable step sizes across tasks and loss scales. Empirically, DAPS trains on high-dimensional datasets (e.g., ImageNet) with superior reconstruction quality, while offering (i) explicit entropy/bit-rate control via β, (ii) stochastic discrete latents amenable to downstream search, and (iii) stable training behavior.

2 Problem Setting and Related Work

We seek to maximize the Evidence Lower Bound (ELBO) of the data likelihood $p(\mathbf{x})$, as introduced in the VAE (Kingma & Welling, 2013; Rezende et al., 2014), but using a discrete latent variable. As in the continuous case, the ELBO is given by the likelihood of the reconstruction minus the KL divergence between the approximate latent posterior and latent prior:

$$\mathcal{L}^{\text{ELBO}}_{\boldsymbol{\phi},\boldsymbol{\theta}} = \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{z}|\mathbf{x}}\Big[\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})\Big] - \beta D_{\text{KL}}\Big(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z})\Big), \tag{1}$$

where the generative model $p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})$ and the recognition model $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ are neural networks parameterized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. The expectations are computed by first sampling $\mathbf{x}$ from the dataset $\mathbf{X}$ and then sampling $\mathbf{z}$ from the recognition model $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

In the continuous setting, we can choose a Gaussian recognition model and reparameterize $\mathbf{z}$ to estimate the gradient of the expectation with respect to $\boldsymbol{\theta}$ for gradient-based optimization. However, in the discrete setting, exact differentiable reparameterization is not possible. To address this challenge, prior work has proposed different approaches based on approximate reparameterization, vector quantization or gradient-free optimization.

Approximate Discrete Reparameterization. Methods based on approximate reparameterization typically apply the Gumbel-Softmax trick, an approximation of the exact but non-differentiable Gumbel-Max reparameterization, that was originally proposed by Jang et al. (2016) and applied to the binary MNIST dataset. Given $\mathbf{x} \sim p(\mathbf{x})$, let $\boldsymbol{\ell}$ represent the unnormalized class probabilities such that $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \text{softmax}(q_{\boldsymbol{\theta}}(\boldsymbol{\ell}|\mathbf{x}))$. Reparameterizing the latent variable can be performed by first sampling Gumbel noise: $\mathbf{g} = -\log(-\log(\mathbf{u}+\epsilon)+\epsilon)$, where $\mathbf{u} \sim \text{Uniform}(\mathbf{0}, \mathbf{1})$ and ϵ is arbitrarily close to zero. Whereas the Gumbel-Max trick would exploit that $z_{\text{hard}} = \arg\max_i \ell_i + g_i$ follows the desired distribution, $z_{\text{hard}} \sim q(z|\mathbf{x})$, the Gumbel-Softmax computes $\mathbf{z}_{\text{soft}} = \text{softmax}((\boldsymbol{\ell}+\mathbf{g})/\tau)$. The temperature τ is a hyperparameter that needs to be carefully chosen to trade off the accuracy of the approximation and the variance of the gradient. In order to pass hard assigned labels to the VAE decoder, Jang et al. (2016) combine the Gumbel-Softmax reparameterization with a straight-through estimator, by using the exact but non-differentiable Gumbel-Max reparameterization during the forward pass, and the approximate but differentiable Gumbel-Softmax reparameterization during the backward pass, which can be straightforwardly implemented as $\mathbf{z} := \mathbf{z}_{\text{soft}} + \text{sg}(\mathbf{z}_{\text{hard}} - \mathbf{z}_{\text{soft}})$, where $\mathbf{z}_{\text{hard}}$ refers to the one-hot encoding of z_{hard} and sg prevents the gradient from flowing through its argument. Although the Gumbel-Softmax trick was successfully used by the discrete VAE (dVAE) in DALL-E (Ramesh et al., 2021) to generate high-quality images, its sensitivity to the temperature parameter is a major hurdle when applying it in practice. Furthermore, backpropagating through the soft-assignments increases the memory footprint, and for autoregressive models the backpropagation-through-time can suffer from vanishing gradients. We will compare to Gumbel-Softmax in our experiments, where we also consider GR-MCK (Paulus et al., 2020), a modification that uses Rao-Blackwellization for variance reduction.

Vector Quantization. VAEs based on vector quantization are arguably the most popular models for learning autoencoders with a discrete latent bottleneck. The VQ-VAE encoder first outputs continuous latent vectors and then maps them to their nearest (in terms of Euclidean distance) vector in a parameterized embedding table $e_{\boldsymbol{\psi}}$. This quantization discretizes the bottleneck, but also introduces a non-differentiable argument minimization in the computational graph. To still update the encoder with respect to the reconstruction loss, VQ-VAEs employ straight-through estimation, in a similar

fashion as used when using the Gumbel-Softmax estimator. Namely, it uses the non-differentiable argument minimization during the forward pass, but skips the quantization step during the backward pass, which can be implemented using a stop-gradient operator as $\mathbf{z}_q := \mathbf{z} + \text{sg}(\mathbf{z}_q - \mathbf{z})$, where $\mathbf{z}$ and $\mathbf{z}_q$ are the encoder outputs before and after quantization, respectively. However, we cannot compute the ELBO as in Eq. 1 since we can not obtain a tractable discrete distribution over the latent variable. Instead, the VQ-VAE optimizes a different loss, given by

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\psi}} = \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\mathbf{z}|\mathbf{x}}\Big[-\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z}_q) + \parallel \text{sg}(\mathbf{z}) - e_{\boldsymbol{\psi}}(\mathbf{z}) \parallel_2^2 + \beta \parallel \mathbf{z} - \text{sg}(e_{\boldsymbol{\psi}}(\mathbf{z})) \parallel_2^2 \Big], \tag{2}$$

where $e_{\boldsymbol{\psi}}(\mathbf{z})$ is the nearest embedding vector in the embedding table to $\mathbf{z}$, and $\mathbf{z}_q$ is computed using the straight-through estimator: $\mathbf{z}_q := \mathbf{z} + \text{sg}(e_{\boldsymbol{\psi}}(\mathbf{z}) - \mathbf{z})$. The expectation with respect to $\mathbf{z} \sim q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ equates to a single sample, i.e., the output of the deterministic recognition model. A more recent method for quantization, also based on the straight-through estimator, is FSQ Mentzer et al. (2023). Instead of learning a codebook, FSQ proposes a rounding mechanism. This design choice results in increased parameter efficiency compared to VQ-VAE (due to the small number of levels typically used), while also avoiding the tricks needed to enforce high codebook usage. FSQ is inspired by compression literature, which often yields solutions that minimize entropy, as noted in their work. Similar to the VAE, we focus on generative modeling from compact latent spaces, but in contrast to quantization-based methods, we explicitly regularize for entropy, as staying close to the prior distribution requires more direct control of the encoder entropy. Finally, it is worth noting that the use of heterogeneous FSQ levels can make FSQ more challenging to implement in downstream tasks compared to other discrete VAEs.

Gradient-Free Optimization. Another branch of methods circumvents the challenges of back-propagating through the discrete sampling process by relying on zero-order methods. In the field of reinforcement learning, REINFORCE (Williams, 1992) is a policy search (Deisenroth et al., 2013) method that estimates the policy gradient using the identity $\nabla_\theta \mathbb{E}[R] = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) R_t]$ to obtain an unbiased estimate of the gradient of the expected return R with respect to the policy parameters θ, without requiring to backpropagate through the expectation. Due to the close relation between reinforcement learning and variational inference (Neumann, 2011; Arenz et al., 2018; Levine, 2018), this approach can be straightforwardly applied to optimize the ELBO 1, where the recognition model $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ takes the role of the policy, and the log-likelihood $\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})$ the role of the return (Tucker et al., 2017; Mnih & Gregor, 2014; Gu et al., 2015; Grathwohl et al., 2017). As REINFORCE suffers from high variance, these methods typically employ variance reduction techniques, such as introducing control variates (Mnih & Gregor, 2014) or combinations of gradient estimators (Tucker et al., 2017; Grathwohl et al., 2017).

However, despite these efforts, the gradient variance of these zero-order methods is still large, resulting in limited success compared to approximate reparameterization or vector quantization. However, there have been significant advances in the field of policy search since the introduction of REINFORCE (Williams, 1992) that have been overlooked in the field of discrete VAEs. In particular, trust-regions based on the Kullback-Leibler divergence to the previous policy (Peters et al., 2010; Schulman, 2015) and zero-order natural gradient estimates (Kakade, 2001; Peters & Schaal, 2008; Wierstra et al., 2014) are standard techniques in reinforcement learning that also seem promising for training discrete VAEs. REPS (Peters et al., 2010) combines both ideas by iteratively solving the optimization problem of maximizing the expected return of a non-parameteric policy subject to a trust-region constraint to the previous policy. REPS has been adapted to many problems, including hierarchical control (Daniel et al., 2016), model-based RL (Abdolmaleki et al., 2015), deep RL (Abdolmaleki et al., 2018), and variational inference (Arenz et al., 2018). Methods that are similar in nature to ours include On-Policy Maximum a Posteriori Optimization (V-MPO) (Song et al., 2019) and Supervised Policy Update (SPU) (Vuong et al., 2018). As in REPS, these methods construct a nonparametric target policy by solving a constrained optimization problem and then update the parameterized policy by minimizing its KL divergence from the target policy, resulting in a weighted maximum likelihood objective. Similar to V-MPO and SPU, we use a neural network-based policy. We also incorporate similar techniques from Lower Bound Policy Search (LBPS) (Watson & Peters, 2023), which optimizes the effective sample size (ESS) to adapt the size of the trust region. These methods have been used to solve complex problems such as robot control (Kober et al., 2013) and Atari (Vuong et al., 2018) and have proven effective in high-dimensional continuous action spaces (Abdolmaleki et al., 2018).

3 DISCRETE AUTOENCODING VIA POLICY SEARCH (DAPS)

We will now present our method for optimizing the ELBO (Eq. 1) with respect to expressive discrete encoders, such as transformers, that can capture the correlations between different latent dimensions through auto-regressive sampling. We avoid computationally expensive and high-variance back-propagation through time by using insights from reinforcement learning that enable us to optimize the encoder using weighted maximum likelihood. In the following, we will present the encoder update and the decoder update separately, which are alternated during training.

Maximizing the ELBO with respect to the variational encoder $q(\mathbf{z}|\mathbf{x})$ can be framed as a KL-regularized reinforcement learning problem. Namely, let us consider a Markov Decision Process (MDP) with state space $\mathcal{X}$, action space $\mathcal{Z}$, and reward function $r(\mathbf{z}, \mathbf{x})$, $\mathbf{z} \in \mathcal{Z}$, $\mathbf{x} \in \mathcal{X}$. Maximum entropy reinforcement learning aims to find a policy $q = q(\mathbf{z}|\mathbf{x})$ that maximizes the expected reward,

$$J(q) = \int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) r(\mathbf{z}, \mathbf{x}) \, d\mathbf{x} + \beta H(q(\mathbf{z}|\mathbf{x}), \tag{3}$$

while also maximizing the entropy $H(q(\mathbf{z}|\mathbf{x})$ of the policy. Comparing to Eq. 1, we can observe that this problem is equivalent to maximizing the ELBO with a uniform prior $p(\mathbf{z})$ over the latent, where the policy corresponds to the encoder and the reward corresponds to the reconstruction error. Reinforcement learning methods such as REPS, SPU and V-MPO, consider two policies: A non-parametric particle-based policy $q^*(\mathbf{z}|\mathbf{x})$ that can be optimized in closed-form but cannot be sampled, and a parametric policy $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ that is trained to approximate $q^*(\mathbf{z}|\mathbf{x})$. While we focus on the case where $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is a categorical distribution over discrete actions (or latent codes), and $\mathbf{x}$ is continuous, we note that our methodology also applies to continuous action spaces and discrete state spaces. In the context of variational autoencoding, the state $\mathbf{x}$ can be viewed as the original data, e.g. a high-dimensional image, from which we want to produce a latent code. The state-action distribution is therefore decomposed into a stochastic policy and a stationary state visit distribution given by our fixed dataset: $q(\mathbf{z}, \mathbf{x}) = q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})p(\mathbf{x})$. The policy is an autoregressive model, such that the joint distribution of categorical variables $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is decomposed into a product of conditional distributions: $q(\mathbf{z}|\mathbf{x}) = \Pi_{t=1}^{T} q_{\boldsymbol{\theta}}(z_t|\mathbf{z}_{\{i\,:\,i<t\}}, \mathbf{x})$.

Optimizing the Nonparametric Encoder. We will start by defining a constrained optimization problem for the objective given by Eq. 3. The procedure is based on REPS, where the objective is to maximize the expected reward while satisfying a constraint on the KL divergence between the policy and a prior policy,

$$\begin{aligned} \max_{q} \quad & \int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) A(\mathbf{z}, \mathbf{x}) \, dx + \beta H(q(\mathbf{z}|\mathbf{x})) \\ \text{s.t.} \quad & D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \parallel q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \leq \epsilon_\eta \quad , \quad \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) = 1. \end{aligned} \tag{4}$$

Here, we replaced the reconstruction reward $r(\mathbf{z}, \mathbf{x})$ with an estimate of the advantage function $A(\mathbf{z}, \mathbf{x})$. The advantage function is obtained by subtracting the value function $V(\mathbf{x})$, which we approximate using the soft-maximum over K latent samples, that is,

$$A(\mathbf{z}, \mathbf{x}) = r(\mathbf{z}, \mathbf{x}) - \log \sum_{k=1}^{K} \exp r(\mathbf{z}^k, \mathbf{x}), \;\; \mathbf{z}^k \sim q_\theta(\mathbf{z}|\mathbf{x}).$$

We can use Lagrangian multipliers to convert the constrained optimization problem into an unconstrained one. As shown in Appendix A.1, the optimal solution is given by:

$$q^*(\mathbf{z}|\mathbf{x}) \propto \exp\left(\frac{A(\mathbf{z}, \mathbf{x}) + \eta \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta + \beta}\right) \tag{5}$$

where η is a Lagrangian multiplier. The term η controls the trust region, i.e., how large of a step we take toward the optimal policy, while β controls the policy entropy. While Eq. 5 provides us with a solution for the optimal policy, we can only evaluate it on given particles, and we cannot compute the normalizer, since summing over all actions is intractable when using a high-dimensional latent bottleneck. However, we will now show that we can use Eq. 5 to evaluate importance weights to update the parametric policy using weighted maximum likelihood.

Algorithm 1 DAPS: Discrete Autoencoding via Policy Search

procedure DAPSLOSSFN($\boldsymbol{\theta}, \boldsymbol{\phi}, \eta, \mathbf{x}, \{\mathbf{z}^k\}_{k=1}^K$)
  for $k = 1$ to K **do**
    Categorical$(.\,; \boldsymbol{\ell}_{\boldsymbol{\theta}}^k(\mathbf{x})) \leftarrow \Pi_{t=1}^T q_{\boldsymbol{\theta}}(.|\mathbf{z}_{\{i\,:\,i<t\}}^k, \mathbf{x})$ (Predict posterior parameters: $\boldsymbol{\ell}_{\boldsymbol{\theta}}^k$)
    $q_{\boldsymbol{\theta}}(\mathbf{z}^k|\mathbf{x}) \leftarrow$ Categorical$(\mathbf{z}^k; \boldsymbol{\ell}_{\boldsymbol{\theta}}^k(\mathbf{x}))$ (Evaluate likelihood of $\mathbf{z}^k$, given logits $\boldsymbol{\ell}_{\boldsymbol{\theta}}^k$)
    $\mathcal{N}(.\,; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{z}^k), \boldsymbol{\sigma}_{\boldsymbol{\phi}}^2(\mathbf{z}^k)) \leftarrow p_{\boldsymbol{\phi}}(.\mid \mathbf{z}^k)$ (Predict posterior parameters: $\boldsymbol{\mu}_{\boldsymbol{\phi}}, \boldsymbol{\sigma}^2$)
    $\mathcal{R}^k \leftarrow \log p_{\boldsymbol{\phi}}(\mathbf{x} \mid \mathbf{z}^k; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\mathbf{z}^k), \boldsymbol{\sigma}_{\boldsymbol{\phi}}^2(\mathbf{z}^k))$ (Compute reward)
  $\mathbf{A}(\mathbf{z}, \mathbf{x}) \leftarrow \boldsymbol{\mathcal{R}} - \log \sum_k \exp(\mathcal{R}^k)$ (Compute advantages)
  $\log \mathbf{q}^*(\mathbf{z}|\mathbf{x}) \propto \dfrac{\mathbf{A}(\mathbf{z}, \mathbf{x}) + \eta \log \mathbf{q}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta + \beta}$ (Compute optimal policy)
  $\mathcal{L}(\boldsymbol{\theta}) \leftarrow -\frac{1}{K} \sum_k \text{SG}\left(\frac{q^*(\mathbf{z}^k|\mathbf{x})}{q_{\boldsymbol{\theta}}(\mathbf{z}^k|\mathbf{x})}\right) \log q_{\boldsymbol{\theta}}(\mathbf{z}^k|\mathbf{x})$ (Recognition model loss)
  $\mathcal{L}(\boldsymbol{\phi}) \leftarrow -\frac{1}{K} \sum_k \mathcal{R}^k$ (Generative model loss)
  $\mathcal{L}(\eta) \leftarrow (\widehat{\text{ESS}}_\eta(q^*, q_{\boldsymbol{\theta}}) - \text{ESS}_{\text{target}})^2$ (η loss)
  return $\mathcal{L}(\boldsymbol{\theta}), \mathcal{L}(\boldsymbol{\phi}), \mathcal{L}(\eta)$

Updating the Parametric Encoder. Now that we have the form of the optimal nonparametric policy, we can move our parameterized policy toward it using maximum likelihood, which enables us to train it without requiring us to backpropagate through the autoregressive sampling. However, because we cannot sample from our nonparametric distribution q^*, we have to resort to importance sampling, where we use the current parametric encoder as proposal distribution. Hence, we maximize a weighted maximum likelihood objective,

$$\mathcal{L}(\boldsymbol{\theta}) = \int_{\mathbf{x}} p(\mathbf{x}) D_{\text{KL}}(q^*(\mathbf{z}|\mathbf{x}) \parallel q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))\, d\mathbf{x} \approx -\sum_i \mathbb{E}_{\mathbf{z}\sim q_\theta(\mathbf{z}|\mathbf{x}_i)} \left[w_i \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)\right] + \text{const} \tag{6}$$

where $w_i = q^*(\mathbf{z}|\mathbf{x}_i) q_\theta(\mathbf{z}|\mathbf{x}_i)^{-1}$ are the importance weights, and the constant term is the entropy of q^* which does not affect optimization. However, due to the unknown normalizer of q^*, we can only evaluate the importance weights up to a constant factor, and therefore resort to self-normalized importance weighting using weights $\tilde{w}_i = w_i / \sum_j w_j$. Self-normalized importance weights benefit from a lower variance but introduce a bias to the approximation, which diminishes asymptotically for large sample sizes.

The new objective can then be summarized as maximizing the log-likelihood of our parameterized policy, weighted by q^*. Note that the update for q^* is largely determined by the reward, which is the log likelihood of $\mathbf{x}$ on the generative model. The generative model is fixed during the policy update, and the process for updating it will now be described in more detail.

Decoder Update. The generative model and the recognition model (i.e., the policy) are updated alternately using coordinate descent. The generative model $p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})$ takes in a latent code $\mathbf{z}$ and outputs parameters of the distribution over $\mathbf{x}$. When optimizing the ELBO with respect to the generative model, it simplifies to

$$\mathcal{L}(\boldsymbol{\phi}) = -\int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})\, d\mathbf{x} \approx -\sum_i \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)} \left[\log p_{\boldsymbol{\phi}}(\mathbf{x}_i|\mathbf{z})\right]. \tag{7}$$

The approximations on the last line of Eq. 7 and Eq. 6 are a result of using Monte Carlo integration with randomly sampled indices from the dataset. Note that this objective corresponds to the standard loss of the VAE in Eq. 1. However, because we do not reparameterize the latent variable, the update is performed independently from the policy update, in contrast to VAEs, where both the generative model and recognition model are updated in one backward pass. Hence, our generative model update does not depend on $\boldsymbol{\theta}$, and the KL divergence to the latent prior can be dropped.

Step Size Adaptation using Effective Sample Size. The multiplier η in Eq. 5 controls the trust region size, i.e. the step size from the parametric policy q_θ toward the nonparametric target q^*. To adapt η automatically, we follow prior work using the *effective sample size* (ESS) (Maia Polo & Vicente, 2023; Metelli et al., 2020; Watson & Peters, 2023), which provides a tractable proxy for

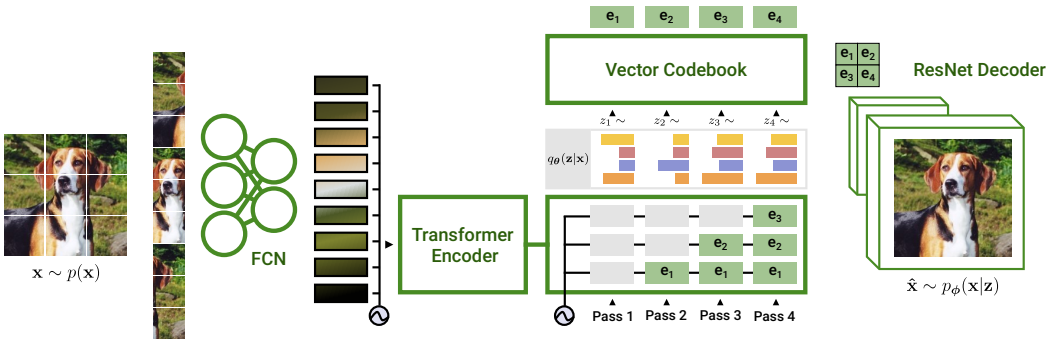


Figure 1: Overview of DAPS. **Left:** Images are split into patches and embedded by a feed-forward network before entering the encoder. **Middle:** The decoder generates latent sequences autoregressively: each step conditions on the encoder output and previously sampled embeddings (via causal masking, shown in green). **Right:** The generative model maps the latent embeddings back to the reconstruction distribution.

the order-2 Rényi divergence. We update η so that the minibatch ESS matches a desired target level $\mathrm{ESS}_{\text{target}}$:

$$\widehat{\mathrm{ESS}}_\eta = \frac{1}{N}\sum_{i=1}^{N} \frac{\left(\sum_{k=1}^{K} w_{ik}\right)^2}{\sum_{k=1}^{K} w_{ik}^2}, \qquad w_{ik} = \frac{q^*(z_i^k \mid x_i; \eta)}{q_\theta(z_i^k \mid x_i)}. \tag{8}$$

Further details and the connection to Rényi divergences are provided in Appendix **??**. In practice, η is treated as a trainable parameter and updated with SGD on $(\widehat{\mathrm{ESS}}_\eta - \mathrm{ESS}_{\text{target}})^2$. We found $\mathrm{ESS}_{\text{target}} \in [K/4,\, 3K/4]$ to yield stable convergence. During training, we observe that η decays smoothly over time, resulting in smoothly decreasing step sizes during optimization, while accounting for the task-specific scale of the reconstruction loss.

4 Experiments

Datasets. We evaluate on four domains of increasing scale and complexity. We will use the term *block size* to denote the number of indices in the latent code (i.e., the maximum length of the transformer decoder), and the term *vocab size* to denote the number of possible values each index can take (i.e., the number of embeddings in the VQ-VAE codebook). The datasets are as follows. **MNIST**: 28×28 (binary) with a 64-bit bottleneck (vocab size 256; block size 8). **CIFAR-10**: 32×32 with a 576-bit bottleneck (vocab size 512; block size 64). **ImageNet-256**: 256×256 with a 10,240-bit bottleneck (vocab size 1,024; block size 1,024). **LAFAN**: motion dataset (∼4.6h expressive human motion, 5 subjects) with a 640-bit bottleneck (vocab size 1,024; block size 64). For LAFAN, motions are retargeted to the Unitree H1 robot and encoded as 32×127 tensors of relative poses. Across datasets, we enforce the same bottleneck size for all methods to ensure fair comparison.

Architectures. Our generative models follow prior work: a 512-unit MLP for MNIST (Kingma & Welling, 2013), and a ResNet decoder for CIFAR/ImageNet (with additionally a learnable variance clipped to $[0.01, 1.0]$) (Van Den Oord et al., 2017). The recognition model is a vision transformer (Alexey, 2020), enabling either autoregressive sampling (non-VQ methods) or direct encoding of codebook vectors (VQ methods) as depicted in Appendix F. For completeness, we also tested the ResNet encoder from (Van Den Oord et al., 2017) in the VQ setting, but observed no advantage. For LAFAN, both recognition and generative models are transformer-based (See Appendix G).

Training. All methods are trained with identical batch sizes and numbers of optimizer steps per dataset. We use Adam with weight decay and a base learning rate of 3×10^{-4}, applying cosine decay when appropriate. For MNIST and CIFAR, we train for 250k steps with a batch size of 256 on an RTX-3090. For LAFAN, we also train for 250k steps with a batch size of 256 on an A100-40GB. For ImageNet, we run for 300k steps with a batch size of 64 on an A100-80GB. Each experiment is repeated with 10 random seeds on MNIST, CIFAR, and LAFAN, and with 5 seeds on ImageNet

Figure 2: ImageNet 256 validation reconstructions. Top to bottom: DAPS, FSQ, Groundtruth.

due to the higher cost. In practice, we find that Gumbel Softmax is unstable at the base learning rate, requiring per-dataset tuning. Because of this instability, we exclude Gumbel Softmax from ImageNet experiments. In contrast, GR-MCK remains stable in the ImageNet setting, though it requires roughly twice the GPU resources.

Baselines. We benchmark DAPS against a range of discrete latent variable models: (1) Gumbel-Softmax, (2) GR-MCK, (3) FSQ, and (4) VQ-VAE. For completeness, we also include the Gaussian VAE (equipped with the same latent capacity) and a PPO-style autoencoding baseline, but omit them from larger-scale experiments due to their substantially weaker performance. For each method, we perform a Bayesian sweep over hyperparameters (listed in Figure 11), centered around the default values used in the original works.

5 Results

We now turn to an empirical evaluation of DAPS. This section is structured as follows: (i) image datasets (MNIST, CIFAR-10, ImageNet), (ii) trajectory datasets (LAFAN), (iii) ablations and hyperparameter sensitivity, and (iv) downstream robotics/control applications.

MNIST. DAPS achieves comparable performance to baseline methods and obtains the lowest L2 train loss when using a final β of 0.01. When encouraging higher entropy (final β of 1.0), DAPS attains a higher validation ELBO than other methods (closely followed by GR-MCK). DAPS also exhibits strong downstream performance when using the latent space to generate images, given a label (see Appendix 13). Compared to quantization-based methods, DAPS generates higher quality digits when decoding samples from a uniform latent prior distribution, which we attribute to the ELBO-based objective (see Appendix 14).

CIFAR-10. As can be seen in Table 1, DAPS outperforms baseline methods on the metrics of interest. Compared to non-VQ-based methods, DAPS quickly and stably attains a high ELBO, without requiring careful scheduling (unlike Gumbel Softmax). Compared to VQ-based methods, DAPS achieves a higher reconstruction likelihood. The FID score is not ideal for CIFAR, as it is tailored for ImageNet; the small increase for VQ-based methods early in training is likely an artifact of this.

ImageNet. We evaluate DAPS on the ImageNet-256 dataset, demonstrating its ability to reconstruct high-dimensional data using a low-bit bottleneck. DAPS significantly outperforms traditional methods in terms of reconstruction quality while maintaining high bit efficiency. When comparing DAPS to FSQ, VQVAE, and GR-MCK on ImageNet-256 validation, DAPS produces high-quality reconstructions, as shown in Fig. 2. DAPS exhibits a higher log likelihood and a lower FID score compared to baseline methods.

Figure 3: A latent space sequence for hopping is decoded by DAPS and fed through inverse kinematics, without physics simulation.

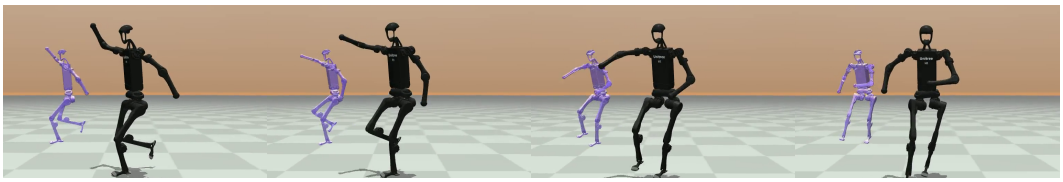


Figure 4: A latent space sequence for dancing is decoded by a reinforcement learning policy and executed in a physics simulator.

LAFAN. We also evaluate DAPS on a LAFAN dataset for the Unitree H1 robot provided in LocoMuJoCo (Al-Hafez et al., 2023), which contains human-to-robot retargeted motion trajectories with inherent temporal dependencies and multimodal characteristics. Quantitatively, DAPS excels at reconstructing this data, outperforming baseline methods in terms of log likelihood and L2 loss. DAPS generates coherent and high-quality motion trajectories, aligning well with the true behaviors in LAFAN (see the supplementary videos). This makes DAPS amenable to downstream robotics tasks such as hierarchical control, where a high-level policy can guide a low-level policy by specifying a desired trajectory, given a context. A reconstructed motion is shown in Figure 3.

Downstream Task: Goal-Conditioned Robot Control. We construct a dataset of motion segments paired with contextual signals and targets. A high-level policy is trained to process a history of relative body poses while conditioning on two context signals: (i) a desired future COM velocity and (ii) a text description of the motion. The policy encodes these inputs into the DAPS latent space, producing a compact summary of the intended future motion that steers a low-level policy. Concretely, the high-level policy is trained to autoregressively generate discrete latents in a distillation setup similar to PoseGPT (Lucas et al., 2022). These latents effectively replace raw motion references, providing a command space—defined by text and COM trajectories—that guides a DeepMimic-trained low-level motion imitation policy. Supplementary videos illustrate the resulting robot behaviors and highlight the utility of DAPS in downstream control tasks. An example motion is shown in Figure 4.

Ablations. We study the two main hyperparameters of DAPS—β and the ESS target—to evaluate the method's sensitivity to their values. Specifically, we consider a grid with $\beta \in \{0.01, 0.1, 1.0\}$ and ESS target $\in \{K/4, K/2, 3K/4\}$ on CIFAR-10. Our results show that annealing β yields the strongest performance (see Appendix C). Intuitively, this schedule promotes exploration early in training and gradually shifts the focus toward high-quality reconstructions later on. In contrast, DAPS shows little sensitivity to the choice of ESS target. This parameter primarily influences the adaptation of η during training to satisfy the entropy objective. Overall, these findings highlight that β is the more critical hyperparameter for performance, while the ESS target has a comparatively minor effect, underscoring the robustness of DAPS to its setting.

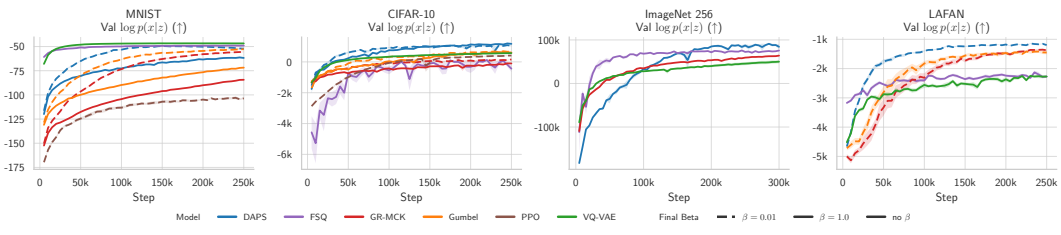


Figure 5: A comparison of reward curves (i.e, $\log p(\mathbf{x}|\mathbf{z})$) on validation sets throughout training.

Dataset	Metric	DAPS		Gumbel		GR-MCK		FSQ	VQ-VAE	PPO
	β	1.0	0.01	1.0	0.01	1.0	0.01			0.01
MNIST (64 bits)	β-ELBO	-82.74	-49.49	-88.46	-52.80	-96.21	-55.58	–	–	-100.97
	$\log p$	-60.80	-49.08	-71.62	-52.45	-84.30	-55.33	-48.81	-46.61	-100.53
	L2	4.18	3.69	4.58	3.85	5.01	3.98	3.73	3.63	5.39
CIFAR-10 (576 bits)	β-ELBO	857.77	1185.51	405.18	704.92	-211.33	217.45	–	–	416.31
	$\log p$	1231.30	1189.37	666.85	707.87	-54.03	219.76	227.13	572.61	419.76
	FID	158.61	157.27	172.68	169.87	184.65	179.88	169.70	162.89	176.29
ImageNet (1.28 KB)	β-ELBO	87.0k	–	–	–	60.7k	–	–	–	–
	$\log p$	93.6k	–	–	–	64.1k	–	76.58k	51.65k	–
	FID	48.65	–	–	–	73.21	–	60.48	76.78	–
LAFAN (640 bits)	β-ELBO	–	-1124.67	–	-1350.55	–	-1290.98	–	–	–
	$\log p$	–	-1120.86	–	-1348.48	–	-1288.77	-1967.92	-2088.26	–
	L2	–	10.19	–	11.80	–	12.14	12.93	13.33	–

Table 1: Quantitative comparison of all methods and datasets.

6 Discussion

Overall, DAPS provides state-of-the-art performance in reconstructing high-dimensional datasets, both for images and sequential motion data, outperforming existing methods across key metrics. The scalability of DAPS, especially with respect to high-dimensional and sequential data, makes it a strong candidate for a variety of generative tasks. Our approach to discrete variational autoencoding via policy search offers a viable alternative to traditional methods like Gumbel-Softmax and VQ-VAE, particularly when dealing with high-dimensional datasets. Using a standard set of VAE hyperparameters, the main parameters for DAPS are the desired effective sample size and β. This makes the training process comparable to that of VQ-VAE, which requires the specification of a commitment coefficient. Both methods introduce hyperparameters that are not highly sensitive, making them relatively easy to tune. While autoregressive discrete sampling allows for expressive multimodal latent generation, it introduces a computational bottleneck, particularly as the sequence length increases. This limits the scalability of both DAPS and Gumbel-Softmax when compared to traditional VAEs and VQ-VAEs. However, DAPS avoids backpropagating through autoregressive sampling, allowing it to scale to sequence lengths of 1,024 and likely beyond. This provides a significant advantage in terms of computational efficiency and scalability, especially on large datasets like ImageNet. Our formulation also allows for fine-grained control over the tradeoff between compression and generalization through the use of β in the ELBO. By adjusting β, we can effectively manage the tradeoff between generating high-quality reconstructions and generalization. This flexibility is a key strength of DAPS over other methods, such as FSQ and VQ-VAE, which do not allow for easy control of entropy. Furthermore, our approach demonstrates stable training across seeds, which is critical for real-world applications.

7 Conclusion

We introduced DAPS, a policy-search-based framework for training discrete variational autoencoders without relying on reparameterization techniques. Our method integrates reinforcement learning concepts to optimize the variational encoder using weighted maximum likelihood, providing both scalability and efficiency. Experiments on high-dimensional datasets such as ImageNet and LAFAN demonstrate DAPS' superior performance in reconstruction quality and its ability to generalize well across different tasks. By combining insights from policy search with variational autoencoding, DAPS opens the door to the efficient use of discrete latent bottlenecks for large-scale generative modeling. Its flexibility in balancing compression and generalization, along with its ability to handle complex data like images and motion trajectories, positions DAPS as a powerful tool for both generative modeling and downstream control tasks. This work also paves the way for further exploration of policy-driven techniques in the optimization of generative models.

REFERENCES

Abbas Abdolmaleki, Rudolf Lioutikov, Jan Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. *Advances in Neural Information Processing Systems*, 28, 2015.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Firas Al-Hafez, Guoping Zhao, Jan Peters, and Davide Tateo. LocoMuJoCo: A comprehensive imitation learning benchmark for locomotion. In *6th Robot Learning Workshop, NeurIPS*, 2023.

Dosovitskiy Alexey. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*, 2020.

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

Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 17(93):1–50, 2016.

Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. *arXiv preprint arXiv:1511.05176*, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Sham M Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Thomas Lucas, Fabien Baradel, Philippe Weinzaepfel, and Grégory Rogez. Posegpt: Quantization-based 3d human motion generation and forecasting. In *European Conference on Computer Vision*, pp. 417–435. Springer, 2022.

Felipe Maia Polo and Renato Vicente. Effective sample size, dimensionality, and generalization in covariate shift adaptation. *Neural Computing and Applications*, 35(25):18187–18199, 2023.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

Alberto Maria Metelli, Matteo Papini, Nico Montali, and Marcello Restelli. Importance sampling techniques for policy optimization. *Journal of Machine Learning Research*, 21(141):1–75, 2020.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pp. 1791–1799. PMLR, 2014.

Gerhard Neumann. Variational inference for policy search in changing situations. 2011.

Max B Paulus, Chris J Maddison, and Andreas Krause. Rao-blackwellizing the straight-through gumbel-softmax gradient estimator. *arXiv preprint arXiv:2010.04838*, 2020.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/rezende14.html.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

H Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, et al. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Quan Vuong, Yiming Zhang, and Keith W Ross. Supervised policy update for deep reinforcement learning. *arXiv preprint arXiv:1805.11706*, 2018.

Joe Watson and Jan Peters. Inferring smooth control: Monte carlo posterior policy iteration with gaussian processes. In *Conference on Robot Learning*, pp. 67–79. PMLR, 2023.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

APPENDIX

A DAPS DERIVATIONS

A.1 SOLVING THE CONSTRAINED OPTIMIZATION PROBLEM

We wish to solve the following constrained optimization problem:

$$\begin{aligned} \max_{q} \quad & \int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) A(\mathbf{z}, \mathbf{x})\, dx + \beta \mathcal{H}(q(\mathbf{z}|\mathbf{x})) \\ \text{s.t.} \quad & D_{\text{KL}}(q(\mathbf{z}|\mathbf{x}) \parallel q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \leq \epsilon_\eta, \quad \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) = 1 \end{aligned} \tag{9}$$

Using Lagrangian multipliers, we can convert this into an unconstrained optimization problem:

$$\begin{aligned} \mathcal{J}(q(\mathbf{z}|\mathbf{x}), \eta, \lambda) = & \int_{\mathbf{x}} p(\mathbf{x}) \sum_{o} q(\mathbf{z}|\mathbf{x}) A(\mathbf{x}, \mathbf{z}) d\mathbf{x} + \beta \mathcal{H}(q(\mathbf{z}|\mathbf{x})) \\ & + \eta \Big(\epsilon_\eta - \int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{q(\mathbf{z}|\mathbf{x}) p(\mathbf{x})}{q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) p(\mathbf{x})} d\mathbf{x} \Big) \\ & + \int_{\mathbf{x}} p(\mathbf{x}) \Big[\lambda(\mathbf{x}) (1 - \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x})) \Big] d\mathbf{x} \end{aligned} \tag{10}$$

Differentiating $\mathcal{J}$ with respect to $q(\mathbf{z}|\mathbf{x})$ and setting to zero yields:

$$\begin{aligned} \frac{\delta \mathcal{J}(q, \eta, \lambda)}{\delta q} = & \int_{\mathbf{x}} \sum_{\mathbf{z}} \frac{\delta}{\delta q} \Big[p(\mathbf{x}) q(\mathbf{z}|\mathbf{x}) \Big(\eta \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \\ & - \eta \log q(\mathbf{z}|\mathbf{x}) - \lambda(\mathbf{x}) + A(\mathbf{x}, \mathbf{z}) - \beta \log q(\mathbf{z}|\mathbf{x}) \Big) \Big] d\mathbf{x} \\ 0 \stackrel{!}{=} & \, p(\mathbf{x}) \Big(\eta \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) - \eta \log q(\mathbf{z}|\mathbf{x}) - \eta - \lambda(\mathbf{x}) + A(\mathbf{x}, \mathbf{z}) - \beta \log q(\mathbf{z}|\mathbf{x}) - \beta \Big) \end{aligned} \tag{11}$$

Re-arranging terms gives us the optimal nonparametric target distribution, $q^*(\mathbf{z}|\mathbf{x})$:

$$\begin{aligned} q^*(\mathbf{z}|\mathbf{x}) &= \exp\Big(\frac{A(\mathbf{x}, \mathbf{z}) + \eta \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta + \beta} \Big) \exp\Big(\frac{\eta + \beta + \lambda(\mathbf{x})}{\eta + \beta} \Big)^{-1} \\ &= q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) \exp\Big(\frac{A(\mathbf{x}, \mathbf{z}) - \beta \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta + \beta} \Big) \exp\Big(\frac{\eta + \beta + \lambda(\mathbf{x})}{\eta + \beta} \Big)^{-1}, \end{aligned} \tag{12}$$

whereby re-writing the last line in Eq. 12, we can see that the optimal policy is a posterior distribution with a prior that is the parametric policy $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. This gives us the following dual function:

$$\begin{aligned} \mathcal{G}(\eta, \lambda(\mathbf{x})) = & \int_{\mathbf{x}} p(\mathbf{x}) \lambda(\mathbf{x}) d\mathbf{x} + \eta \epsilon_\eta \\ & + \int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q^*(\mathbf{z}|\mathbf{x}) \Big[\eta \log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) - (\eta + \beta) \log q^*(\mathbf{z}|\mathbf{x}) - \lambda(\mathbf{x}) + A(\mathbf{x}, \mathbf{z}) \Big] d\mathbf{x} \\ = & \int_{\mathbf{x}} p(\mathbf{x}) \lambda(\mathbf{x}) d\mathbf{x} + \eta \epsilon_\eta + (\eta + \beta) \int_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{z}} q^*(\mathbf{z}|\mathbf{x}) d\mathbf{x} \end{aligned} \tag{13}$$

Since the optimal $\lambda(\mathbf{x})$ normalizes $q^*(\mathbf{z}|\mathbf{x})$, we have:

$$\exp\Big(\frac{\eta+\beta+\lambda^*(\mathbf{x})}{\eta+\beta}\Big) = \sum_{\mathbf{z}} \exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big)$$
$$\lambda^*(\mathbf{x}) = (\eta+\beta)\log\sum_{\mathbf{z}} \exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big) - \eta - \beta \tag{14}$$

Plugging this into the dual, we get:

$$\begin{aligned}\mathcal{G}(\eta,\lambda^*(\mathbf{x})) &= \int_{\mathbf{x}} p(\mathbf{x})\Big[(\eta+\beta)\log\sum_{\mathbf{z}}\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big) - \eta - \beta\Big]d\mathbf{x} + \eta\epsilon_\eta \\ &\quad + (\eta+\beta)\int_{\mathbf{x}} p(\mathbf{x})\sum_{\mathbf{z}} q^*(\mathbf{z}|\mathbf{x})d\mathbf{x} \\ &= (\eta+\beta)\int_{\mathbf{x}} p(\mathbf{x})\Big[\log\sum_{\mathbf{z}}\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big)\Big]d\mathbf{x} + \eta\epsilon_\eta\end{aligned} \tag{15}$$

We can now differentiate with respect to η to optimize our stepsize.

$$\frac{\delta\mathcal{G}(\eta,\lambda^*(\mathbf{x}))}{\delta\eta} = \frac{\delta}{\delta\eta}\Big[(\eta+\beta)\int_{\mathbf{x}} p(\mathbf{x})\Big[\log\sum_{\mathbf{z}}\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big)\Big]d\mathbf{x} + \eta\epsilon_\eta\Big] \tag{16}$$

Letting $v = \sum_{\mathbf{z}}\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big)$ and $u = \log v$, we have that $\frac{\delta u}{\delta v} = v^{-1}$ and:

$$\frac{\delta v}{\delta\eta} = \sum_{\mathbf{z}}\Big(\frac{\beta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) - A(\mathbf{x},\mathbf{z})}{(\eta+\beta)^2}\Big)\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big) \tag{17}$$

Applying the chain rule, using $\frac{\delta u}{\delta\eta} = \frac{\delta u}{\delta v}\frac{\delta v}{\delta\eta}$, we get:

$$\begin{aligned}\frac{\delta\mathcal{G}(\eta,\lambda^*(\mathbf{x}))}{\delta\eta} &= \int_{\mathbf{x}} p(\mathbf{x})\Big[\sum_{\mathbf{z}}\Big(\frac{\beta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) - A(\mathbf{x},\mathbf{z})}{\eta+\beta}\Big)\frac{\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big)}{\sum_{\mathbf{z}'}\exp\Big(\frac{A(\mathbf{x},\mathbf{z}')+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}'|\mathbf{x})}{\eta+\beta}\Big)} \\ &\quad + \log\sum_{\mathbf{z}}\exp\Big(\frac{A(\mathbf{x},\mathbf{z})+\eta\log q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\eta+\beta}\Big)\Big]d\mathbf{x} + \epsilon_\eta \\ &= \int_{\mathbf{x}} p(\mathbf{x})\Big[\sum_{\mathbf{z}}\log\frac{q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{\tilde{q}^*(\mathbf{z}|\mathbf{x})}q^*(\mathbf{z}|\mathbf{x}) + \log Z_{q^*}\Big]d\mathbf{x} + \epsilon_\eta \\ &= \epsilon_\eta - \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\Big[D_{\mathrm{KL}}\Big(q^*(\mathbf{z}|\mathbf{x}) \parallel q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\Big)\Big]\end{aligned} \tag{18}$$

It can be seen that computing this value requires computing the partition function, Z_{q^*}, and this requires enumerating over all possible sequences of $\mathbf{z}$. Similarly, the outer summation cannot be easily computed, rendering the optimization of η using the KL divergence difficult in our setting. For this reason, we make use of the Effective Sample Size to optimize η, as described below.

A.2 ESS-based Trust Region

Setup. Given an arbitrary datapoint $\mathbf{x}$, let $q_\theta(\mathbf{z} \mid \mathbf{x})$ be the proposal distribution. Let $\tilde{q}^*(\mathbf{z} \mid \mathbf{x})$ denote the *unnormalized* target from Eq. 5, with partition function

$$Z(\mathbf{x}) \;=\; \sum_{\mathbf{z}} \tilde{q}^*(\mathbf{z} \mid \mathbf{x}), \qquad q^*(\mathbf{z} \mid \mathbf{x}) \;=\; \frac{\tilde{q}^*(\mathbf{z} \mid \mathbf{x})}{Z(\mathbf{x})}.$$

Drawing $\mathbf{z}_{1:K} \sim q_\theta(\cdot \mid \mathbf{x})$, define the normalized importance weights

$$w_k \;=\; \frac{q^*(\mathbf{z}_k \mid \mathbf{x})}{q_\theta(\mathbf{z}_k \mid \mathbf{x})}.$$

Definition The effective sample size (ESS) associated with weights $\{w_k\}_{k=1}^K$ is

$$\widehat{\mathrm{ESS}}_K(\mathbf{x}) \;=\; \frac{\Big(\sum_{k=1}^K w_k\Big)^2}{\sum_{k=1}^K w_k^2}.$$

Scale invariance. If instead we had used the unnormalized target $\tilde{q}^*$, then

$$\tilde{w}_k \;=\; \frac{\tilde{q}^*(\mathbf{z}_k \mid \mathbf{x})}{q_\theta(\mathbf{z}_k \mid \mathbf{x})} = Z(\mathbf{x})\, w_k.$$

The ESS is invariant to such rescaling:

$$\frac{(\sum_k \tilde{w}_k)^2}{\sum_k \tilde{w}_k^2} = \frac{(Z\sum_k w_k)^2}{Z^2\sum_k w_k^2} = \frac{(\sum_k w_k)^2}{\sum_k w_k^2}.$$

Thus one may compute $\widehat{\mathrm{ESS}}_K$ using either q^* or $\tilde{q}^*$.

Normalized ESS ratio. For convenience, we define the *normalized ESS ratio*

$$\widehat{\rho}_K(\mathbf{x}) \;=\; \frac{1}{K}\,\widehat{\mathrm{ESS}}_K(\mathbf{x}) \;=\; \frac{1}{K}\,\frac{(\sum_{k=1}^K w_k)^2}{\sum_{k=1}^K w_k^2} \;=\; \frac{\big(\frac{1}{K}\sum_{k=1}^K w_k\big)^2}{\frac{1}{K}\sum_{k=1}^K w_k^2}.$$

This form makes both numerator and denominator empirical means, so the strong law of large numbers applies directly.

Definition (Rényi–2 divergence). For two distributions p, q with p absolutely continuous w.r.t. q, the order–2 Rényi divergence is

$$D_2(p\|q) \;=\; \log \sum_{\mathbf{z}} \frac{p(\mathbf{z})^2}{q(\mathbf{z})}.$$

Lemma 1 (Population ESS and Rényi–2)**.** *Assume $\mathbb{E}_{q_\theta}[w^2] < \infty$, which holds in the case of finite discrete distributions with full support. Then, as $K \to \infty$,*

$$\widehat{\rho}_K(\mathbf{x}) \;\xrightarrow{\text{a.s.}}\; \rho(\mathbf{x}) = \frac{(\mathbb{E}_{q_\theta}[w])^2}{\mathbb{E}_{q_\theta}[w^2]} = \frac{1}{\mathbb{E}_{q_\theta}[w^2]} = \exp\big(-D_2(q^*\|q_\theta)\big).$$

Proof. By the strong law,

$$\frac{1}{K}\sum_{k=1}^K w_k \;\to\; \mathbb{E}_{q_\theta}[w], \qquad \frac{1}{K}\sum_{k=1}^K w_k^2 \;\to\; \mathbb{E}_{q_\theta}[w^2],$$

almost surely. Since q^* is normalized,

$$\mathbb{E}_{q_\theta}[w] = \sum_{\mathbf{z}} q_\theta(\mathbf{z} \mid \mathbf{x})\,\frac{q^*(\mathbf{z} \mid \mathbf{x})}{q_\theta(\mathbf{z} \mid \mathbf{x})} = 1.$$

Moreover, by the definition of D_2,

$$\mathbb{E}_{q_\theta}[w^2] = \sum_{\mathbf{z}} q_\theta(\mathbf{z} \mid \mathbf{x})\,\Big(\frac{q^*(\mathbf{z} \mid \mathbf{x})}{q_\theta(\mathbf{z} \mid \mathbf{x})}\Big)^2 = \sum_{\mathbf{z}} \frac{q^*(\mathbf{z} \mid \mathbf{x})^2}{q_\theta(\mathbf{z} \mid \mathbf{x})} = \exp\big(D_2(q^*\|q_\theta)\big).$$

Therefore $\rho(\mathbf{x}) = 1/\mathbb{E}[w^2] = \exp(-D_2(q^*\|q_\theta))$. □

Corollary 1 (KL trust region from ESS target)**.** *By monotonicity of Rényi divergences in their order (Van Erven & Harremos, 2014),*

$$\mathrm{KL}(q^*\|q_\theta) = D_1(q^*\|q_\theta) \;\le\; D_2(q^*\|q_\theta).$$

Hence, if $\widehat{\rho}_K(\mathbf{x})$ is adapted to match a target level ρ_{target}, then the update satisfies the KL trust-region bound

$$\mathrm{KL}(q^*\|q_\theta) \;\le\; -\log \rho_{\text{target}}.$$

Remarks. *(i) $\widehat{\rho}_K$ is a biased finite-sample estimator of the population $\rho(\mathbf{x})$, but converges almost surely by the strong law. (ii) In practice, we treat η as a trainable parameter and update it with stochastic gradient descent on $(\widehat{\rho}_K - \rho_{\text{target}})^2$, thereby driving $\widehat{\rho}_K$ toward ρ_{target}.*

B Training Dynamics and Stability

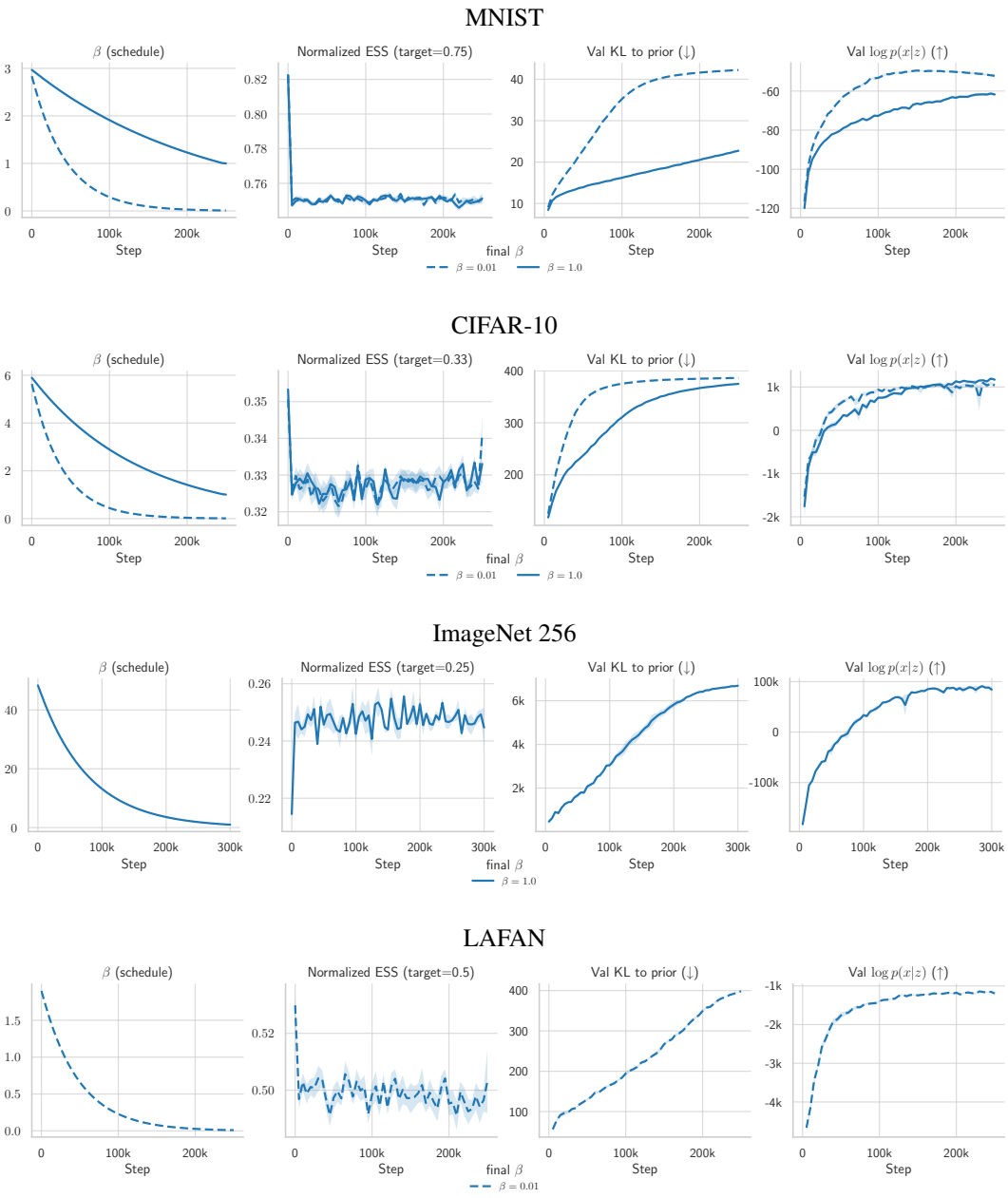


Figure 6: A summary of key metrics for DAPS throughout training.

C Ablations: β and ESS-Target (CIFAR-10)

We scan $\beta \in \{0.1, 1, 5, 10\}$ and ESS-target $\in \{K/4, K/2, 3K/4\}$ and report the metrics below.

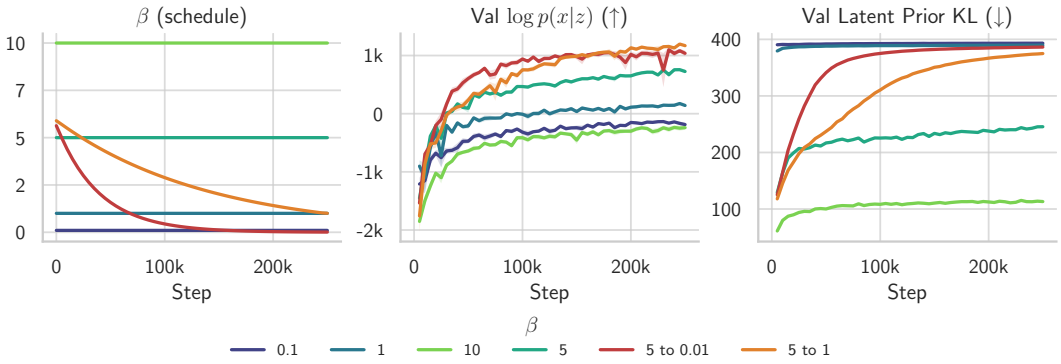


Figure 7: (1) An overview of different β values, averaged across the 3 different ESS targets. (2) The scheduled β curves are averaged over the 10 seeds from the final experiments, which use an ESS target of 0.33.

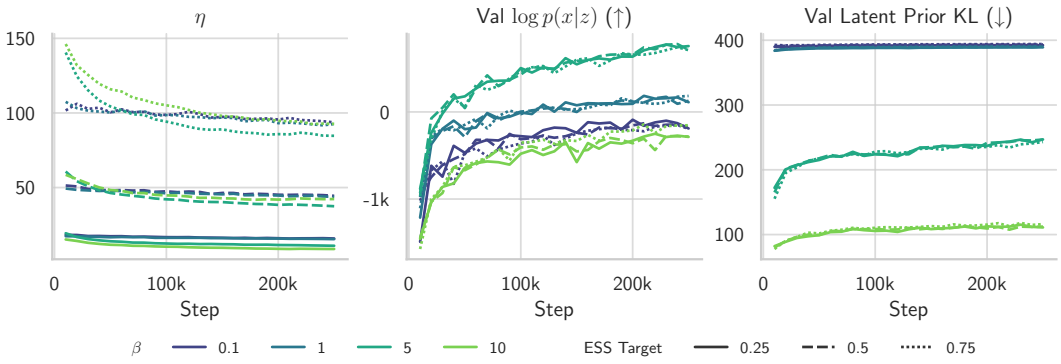


Figure 8: The joint performance of various β and ESS combinations. It can be seen that the choice of ESS target has the desired effect, namely, the attenuation of the η trajectory throughout training.

D Ablation: Number of baseline samples (MNIST)

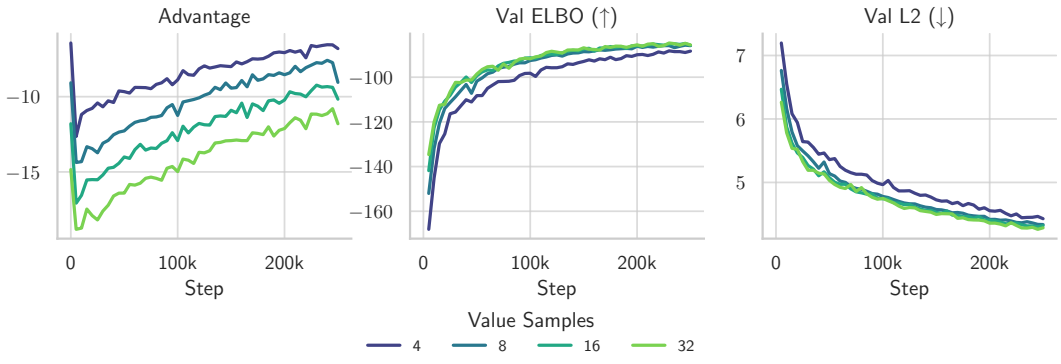


Figure 9: Key metrics w.r.t the number of samples used in the advantage estimation.

E PERFORMANCE SUMMARY

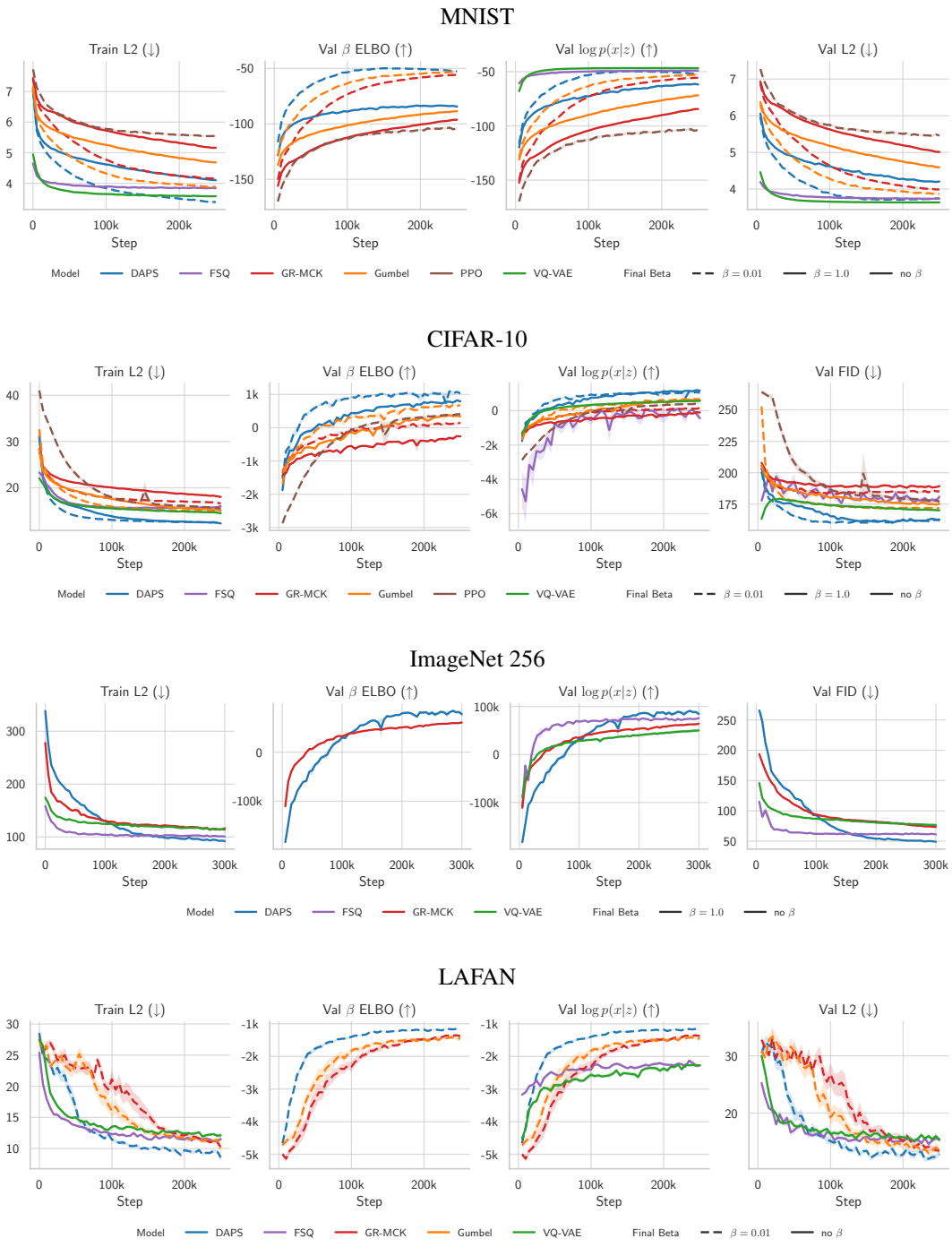
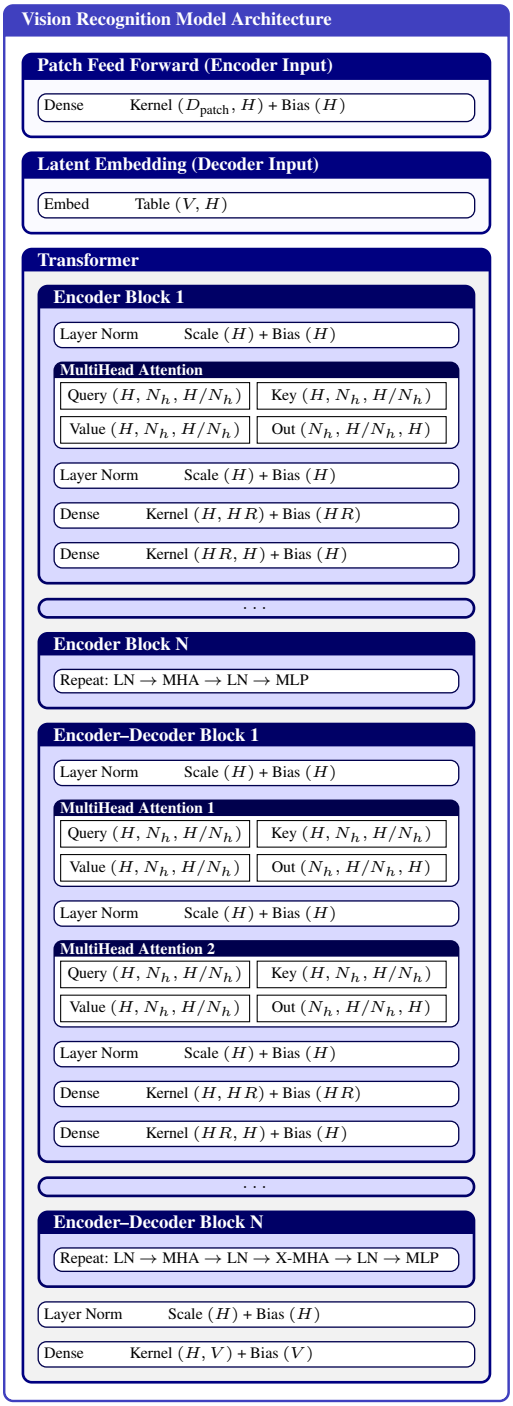
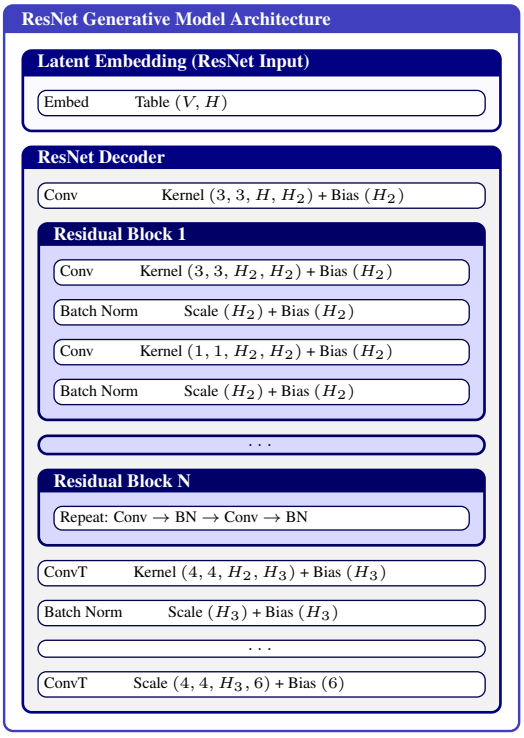


Figure 10: A summary of training curves averaged over all seeds, ± 1 SE.

F VISION MODEL ARCHITECTURE AND PARAMETER SUMMARY



Recognition Model. The recognition model architecture is a standard vision transformer. Cross attention is used for autoregressive-based methods, while self-attention is used for VQ-based methods. We also tried using the ResNet encoder proposed in VQ-VAE, but found no performance benefit. See F.1 and F.2 for the corresponding variable values.



Generative Model. The vision generative model architecture is a standard ResNet decoder. This model takes as input a latent sequence (of length 64 for CIFAR-10, and length $1,024$ for ImageNet) and converts it to a sequence of embeddings using the embedding table. This is then reshaped into a square tensor of embeddings ($8 \times 8 \times 128$ for CIFAR-10, and $32 \times 32 \times 128$ for ImageNet) to be processed by the ResNet. After which, convolutional transpose layers are used to upsample the tensor to the final image dimension, including an additional 3 channels for the RGB variance. See F.1 and F.2 for the corresponding variable values.

F.1 IMAGENET-256 VISION HYPERPARAMETERS

Config (ImageNet-256)

```yaml
model:
  block_size: 1024                    # Latent length
  vocab_size: 1024                    # V
  embed_dim: 128                      # H

data:
  image_size: 256
  num_patches: 1024                   # 32x32 patches

recog_attn:                           # Self Attention (VQ methods)
  transformer:
    num_heads: 4                      # N_h
    num_layers: 2                     # N
    mlp_ratio: 4                      # R
    embed_dim: ${model.embed_dim}
    decoder_block_size: ${model.block_size}
    qkv_dim: ${model.embed_dim}
  patch_ffwd: [dense:${model.embed_dim}]

recog_cross:                          # Cross Attention (non-VQ methods)
  <<: *recog_attn
  transformer:
    <<: *recog_attn.transformer
    grow_target_every: 8              # Autoreg. cache frequency
    encoder_block_size: ${data.num_patches}

generative:
  resnet:
    hidden_dim: 64                    # H_2
    activation: relu
    norm: batch_norm
    residual:
      num_blocks: 2                   # N
      hidden_dim: 64                  # H_2
      activation: relu
      norm: batch_norm
```

Network Hyperparameter Summary. The configuration above lists the architecture-specific hyperparameters used for training ImageNet 256 (shared by all methods). Variable names, indicated by comments on the right-hand side, correspond to the variables in the Vision Transformer Diagram featured in F. The autoregressive sampling used by ELBO-based methods can be sped up through the use of KV caching, as past activations do not need to be recalculated on subsequent forward passes. The granularity of this caching procedure is controlled by the `grow_target_every` attribute, which determines the size of the encoder's cache window.

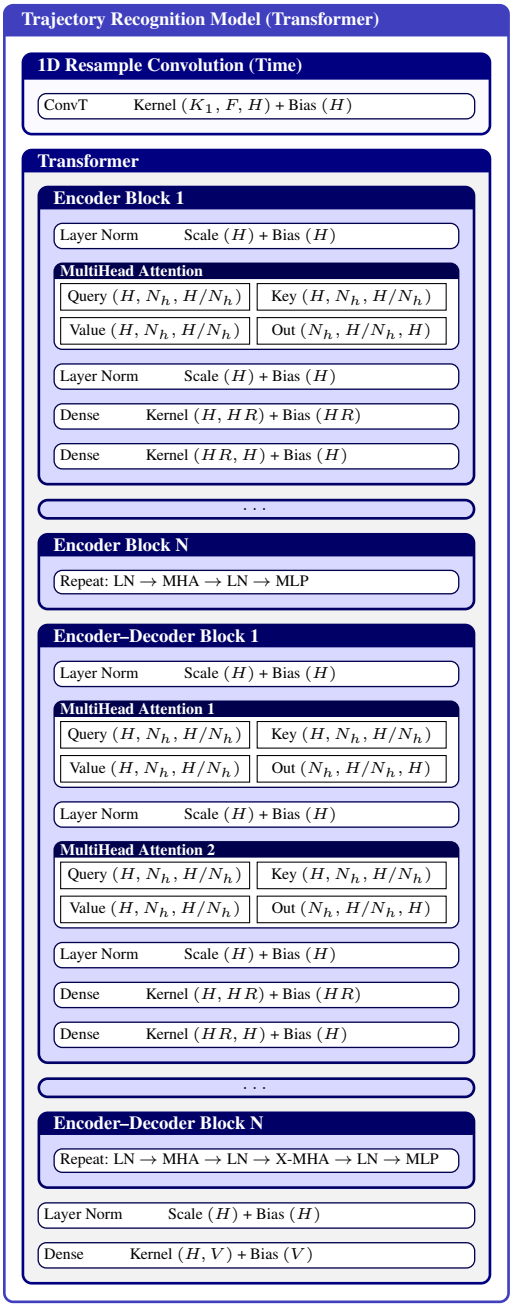
F.2 CIFAR-10 Vision Hyperparameters

Config (CIFAR-10)

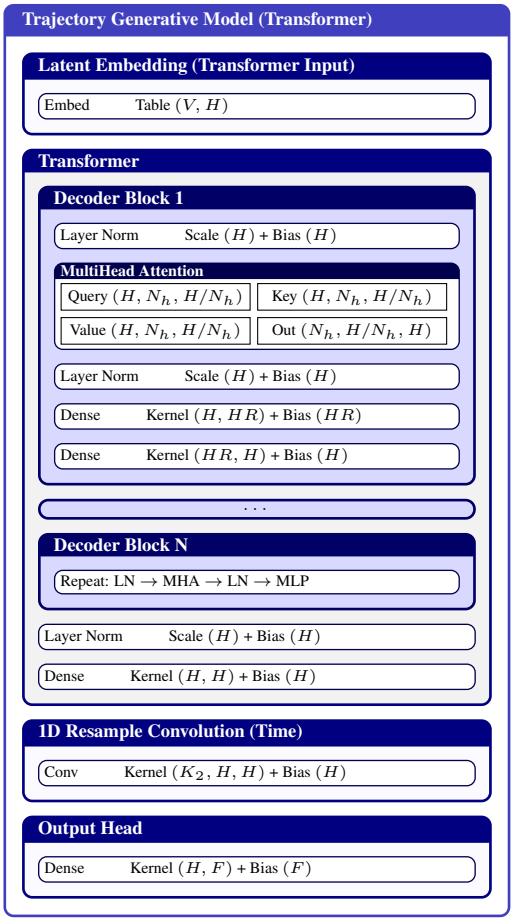
```yaml
model:
  block_size: 64                    # Latent length
  vocab_size: 512                   # V
  embed_dim: 128                    # H

data:
  image_size: 32
  num_patches: 64                   # 8x8 patches

recog_attn:                         # Self Attention (VQ methods)
  transformer:
    num_heads: 4                    # N_h
    num_layers: 2                   # N
    mlp_ratio: 4                    # R
    embed_dim: ${model.embed_dim}
    decoder_block_size: ${model.block_size}
    qkv_dim: ${model.embed_dim}
  patch_ffwd: [dense:${model.embed_dim}]

recog_cross:                        # Cross Attention (non-VQ methods)
  <<: *recog_attn
  transformer:
    <<: *recog_attn.transformer
    grow_target_every: 8            # Autoreg. cache frequency
    encoder_block_size: ${data.num_patches}

generative:
  resnet:
    hidden_dim: 64                  # H_2
    activation: relu
    norm: batch_norm
    residual:
      num_blocks: 2                 # N
      hidden_dim: 64                # H_2
      activation: relu
      norm: batch_norm
```

Network Hyperparameter Summary. The configuration above lists the architecture-specific hyperparameters used for training CIFAR-10 (shared by all methods). Variable names, indicated by comments on the right-hand side, correspond to the variables in the Vision Transformer Diagram featured in F. This design follows the architectures presented in Van Den Oord et al. (2017) and Alexey (2020).

G TRAJECTORY MODEL ARCHITECTURE AND PARAMETER SUMMARY



Recognition Model. The trajectory recognition model architecture shares a similar structure to the vision recognition model. Instead of using an image patcher (as in the ViT), we use a 1D convolutional network to embed the trajectory sequence. Cross attention is used for autoregressive-based methods, while self-attention is used for VQ-based methods.



Generative Model. The trajectory generative model architecture is similar in symmetry to the trajectory recognition model. After processing the embedded latent sequence, the sequence is resampled by a 1D convolutional network to reach the desired trajectory segment length. The result is a continuous trajectory of length 32 with 127 features per timestep.

G.1 LAFAN TRAJECTORY HYPERPARAMETERS

Config (Trajectory: LAFAN)

```yaml
model:
  block_size: 64                          # Latent length
  vocab_size: 1024                        # V
  embed_dim: 128                          # H

trajectory:
  length: 32                              # Segment length
  feature_dim: 127                        # F

traj_recog_attn:                          # Self Attention (VQ methods)
  resample_conv1d:
    output_length: ${model.block_size}
    output_channels: ${model.embed_dim}
    stride: 2
    kernel_size: 4                        # K_1
  transformer:
    num_heads: 4                          # N_h
    num_layers: 2                         # N
    mlp_ratio: 4                          # R
    embed_dim: ${model.embed_dim}
    decoder_block_size: ${model.block_size}
    qkv_dim: ${model.embed_dim}

traj_recog_cross:                         # Cross Attn (non-VQ methods)
  <<: *traj_recog_attn
  transformer:
    <<: *traj_recog_attn.transformer
    grow_target_every: 8                  # Autoreg. cache frequency
    encoder_block_size: ${model.block_size}

traj_generative:                          # Generative model (all
↪  methods)
  transformer:
    <<: *traj_recog_attn.transformer
  resample_conv1d:
    output_length: ${trajectory.length}
    output_channels: ${model.embed_dim}
    stride: 2
    kernel_size: 4                        # K_2
  output_ffwd: [dense:${trajectory.feature_dim}]
```

Network Hyperparameter Summary. The configuration above lists the architecture-specific hyperparameters used for training the LAFAN dataset. Variable names, indicated by comments on the right-hand side, correspond to the variables in the Trajectory Transformer Diagram featured in G. The recognition model and generative models are highly symmetric. These networks reconstruct randomly sampled trajectory segments of length 32 with feature dimension 127. In order to decouple the block size from the length of the trajectory segment, we use a 1D convolutional network to resample the trajectory length to the model's block size. This operation is then reversed at the output of the generative model to achieve the desired segment length.

H MODEL-SPECIFIC HYPERPARAMETERS

MNIST

Parameter	DAPS	Other
Learning rate	3e−4	{3, 3, 3, 3}e−4
Initial β (GR-MCK / Gumbel)	3	3
ESS Target	$3K/4$	–
Number of Baseline Samples (K)	8	–
τ (GR-MCK / Gumbel)	–	1.0 / 1.0
Commitment Coefficient (FSQ)	–	0.25
Levels (FSQ)	–	[8, 6, 5]

CIFAR-10

Parameter	DAPS	Other
Learning rate	3e−4	{3, 3, 3, 3α1}e−4
Initial β (GR-MCK / Gumbel)	6	3 / 6
ESS Target	$K/3$	–
Number of Baseline Samples (K)	8	–
τ (GR-MCK / Gumbel)	–	1.0 to 0.75 / 1.0
Commitment Coefficient (FSQ)	–	0.25
Levels (FSQ)	–	[8, 8, 8]

ImageNet 256

Parameter	DAPS	Other
Learning rate	3e−4	{3, 3, 1}e−4
Initial β (GR-MCK)	50	10
ESS Target	$K/4$	–
Number of Baseline Samples (K)	8	–
τ (GR-MCK)	–	1.0 to 0.75
Commitment Coefficient (FSQ)	–	0.25
Levels (FSQ)	–	[8, 5, 5, 5]

LAFAN

Parameter	DAPS	Other
Learning rate	3e−4	{3, 3, 1α1, 1α1}e−4
Initial β (GR-MCK / Gumbel)	2	2 / 2
ESS Target	$K/2$	–
Number of Baseline Samples (K)	8	–
τ (GR-MCK / Gumbel)	–	1.0 / 1.0
Commitment Coefficient (FSQ)	–	0.5
Levels (FSQ / VQ)	–	[10, 10, 10]

Figure 11: A summary of model-specific hyperparameters.

Note: The order for *Other* is: VQ-VAE, FSQ, GR-MCK, and Gumbel (by default, where applicable). The $x\ \alpha\ y$ notation means a start rate of x and final rate of $xy/10$ using cosine decay.

I Downstream Robotics Experiment Details

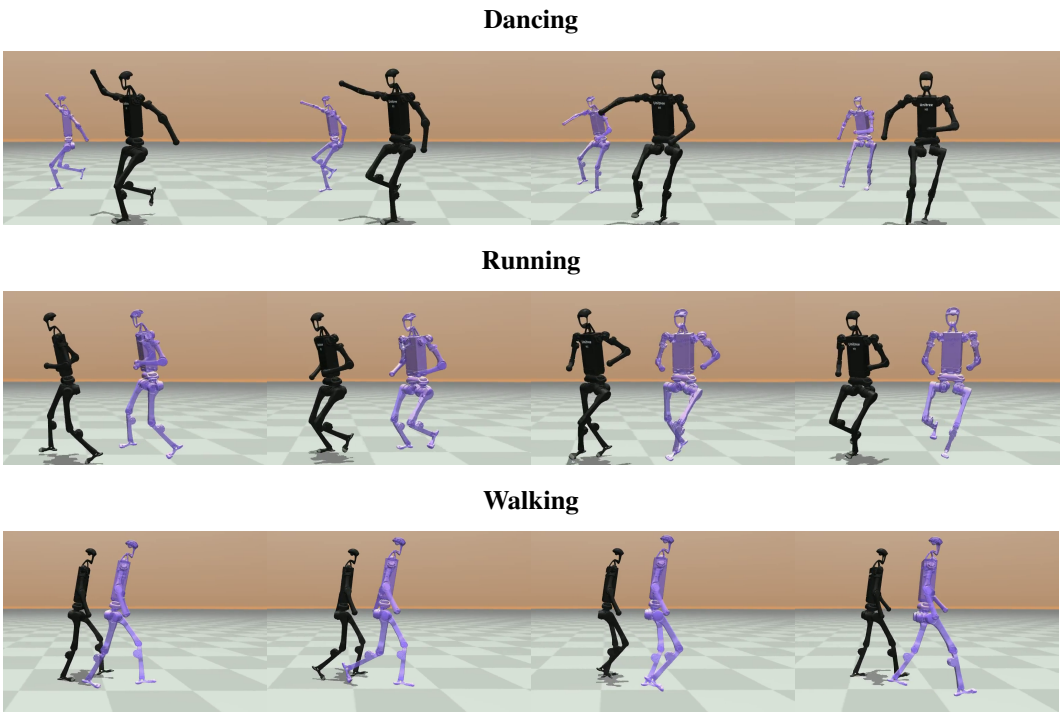


Figure 12: In our downstream experiment, we use different language commands to generate motion styles with DAPS, and use a physical consistent decoder given by an RL policy and a robot simulator to decode the discrete latent representation.

We train a low-level policy with DeepMimic to imitate motions from the LAFAN1 dataset on a Unitree H1 robot simulated in MuJoCo. Unlike the conventional DeepMimic formulation, where motion reference information for the next timestep (site positions, joint angles, and velocities) is explicitly given as a goal to the policy, we provide the policy with a compact embedding of the desired future motion as the goal. We condition the high-level policy—which produces these embeddings—on language information and motion goals. For the former, we use BERT embeddings of a brief description of the motion file. For the latter, we provide the encoder with the last 32 timesteps and future 32 timesteps of the center of mass (COM) trajectory, relative to the robot's base frame. The resulting output is a compact vector of 64 latent indices.

We train the low-level policy conditioned on these embeddings with PPO using the LocoMujoco framework (Al-Hafez et al., 2023). The policy transforms the embedded indices with a 1D convolutional layer, then concatenates the resulting features with standard proprioceptive information: joint angles, joint velocities, IMU information (projected gravity and torso angular velocities), and the previous timestep's action, ultimately feeding these into a multilayer perceptron of size [2048, 1024, 512] with `elu` activations, resulting in a 19-dimensional vector of torques to be applied to each motor of the Unitree H1 robot. The policy is executed at 100 Hz.

J Datasets and Preprocessing

Images. For ImageNet-256, we randomly crop and flip the images. All images are standardized using their empirical mean and standard deviation. CIFAR-10 and MNIST are also standardized similarly.

Trajectories (LAFAN). For the LAFAN dataset, we standardize the data online using the Welford algorithm (for numerically stable variance estimation). Each motion sequence is normalized to have zero mean and unit variance along each feature.

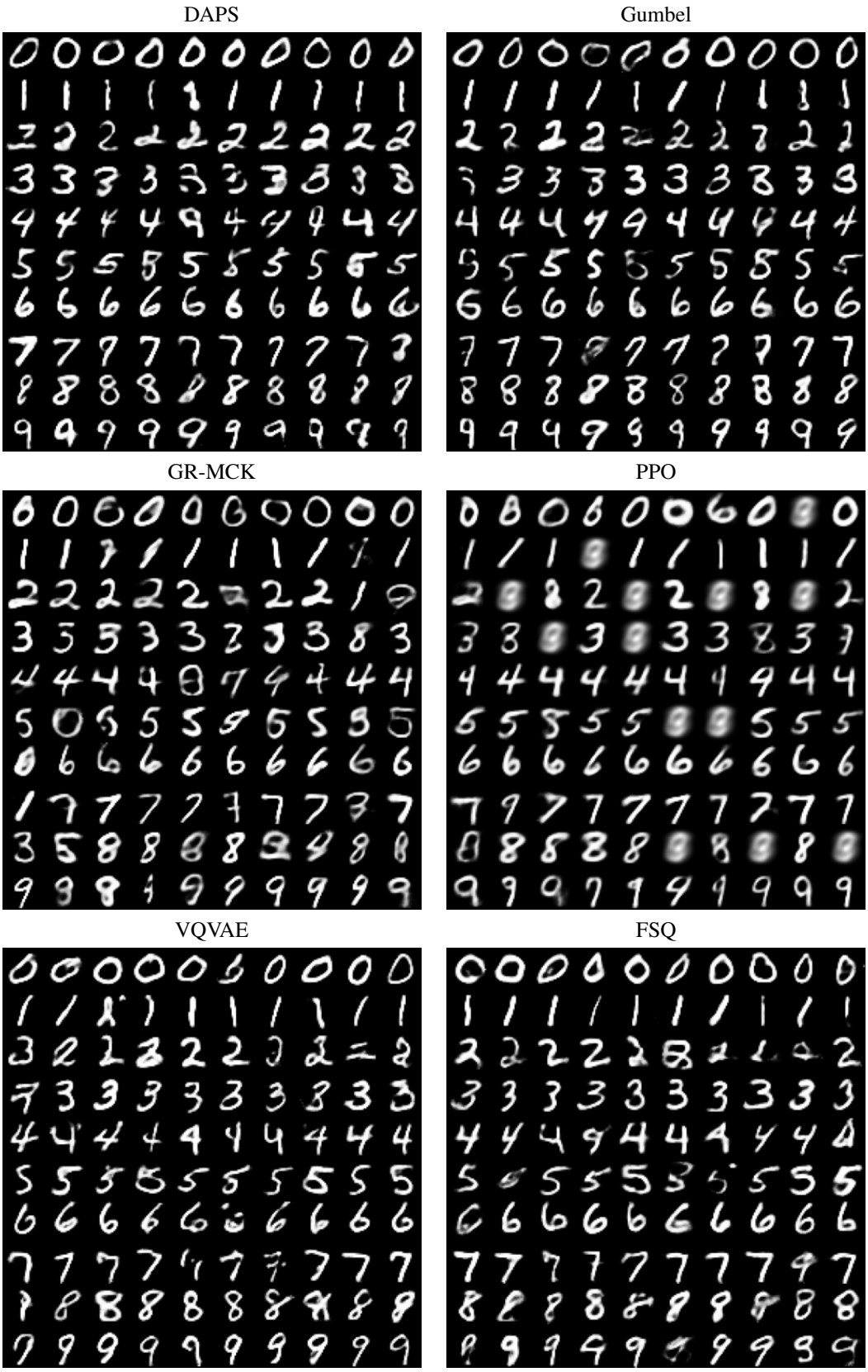


Figure 13: Label-conditioned samples from downstream model using discrete flow matching.

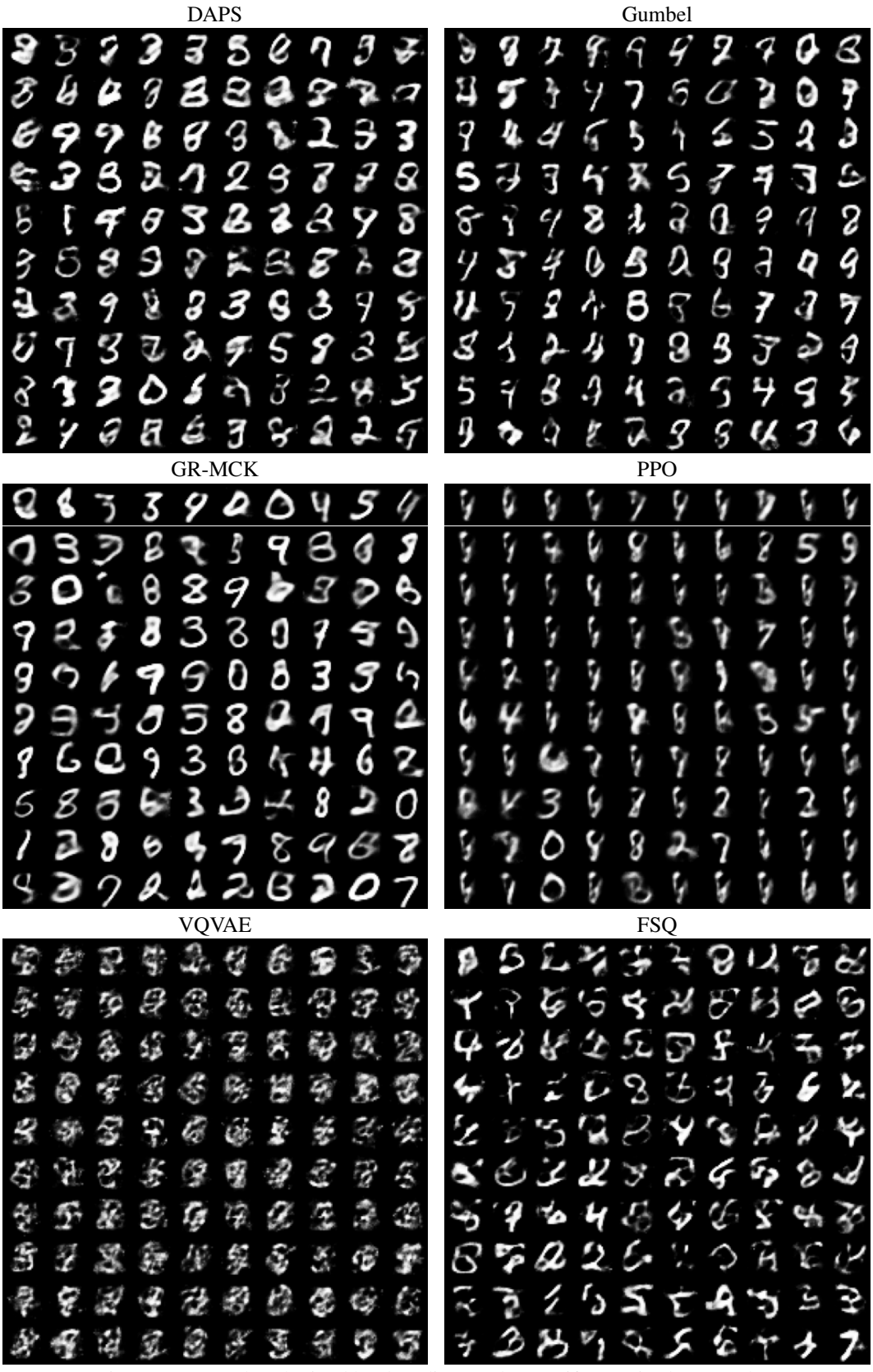


Figure 14: Samples obtained by decoding random, uniform prior latent codes.

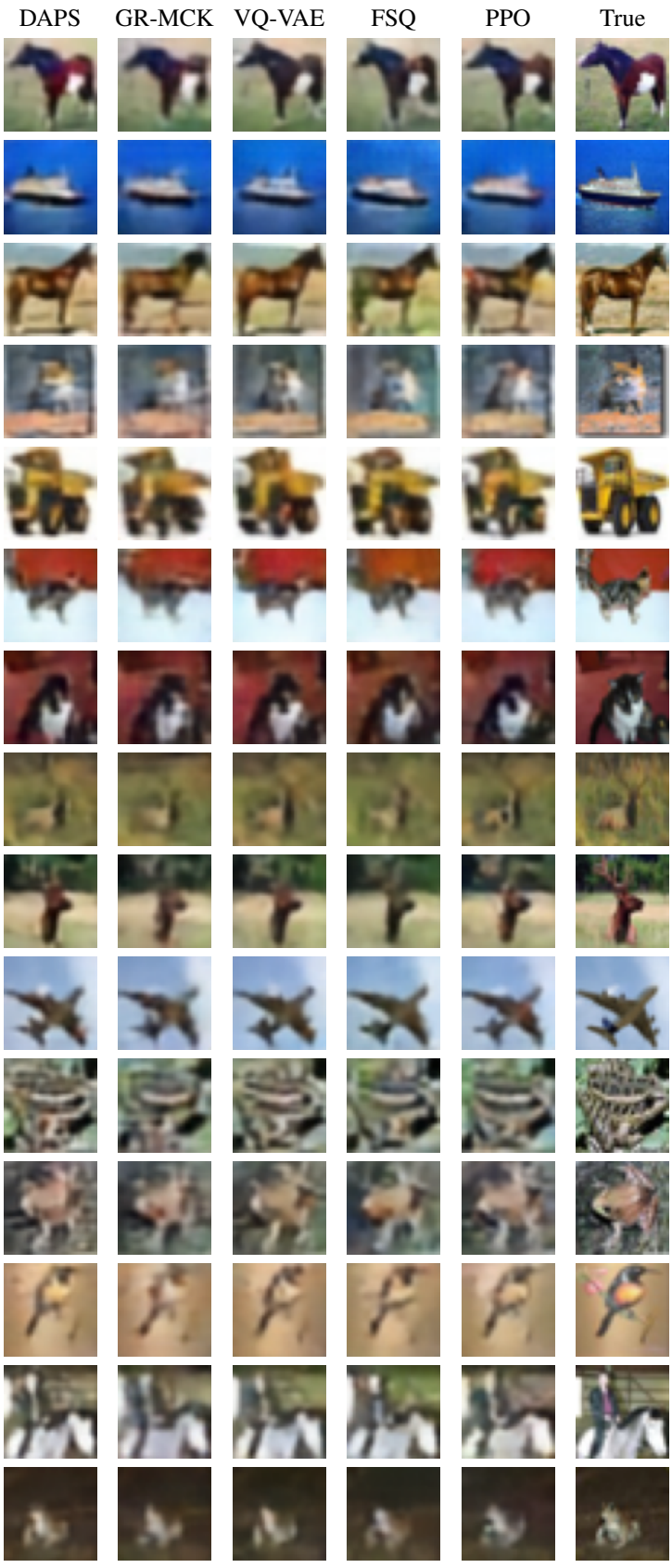


Figure 15: CIFAR validation reconstructions vs. ground truth for six methods.

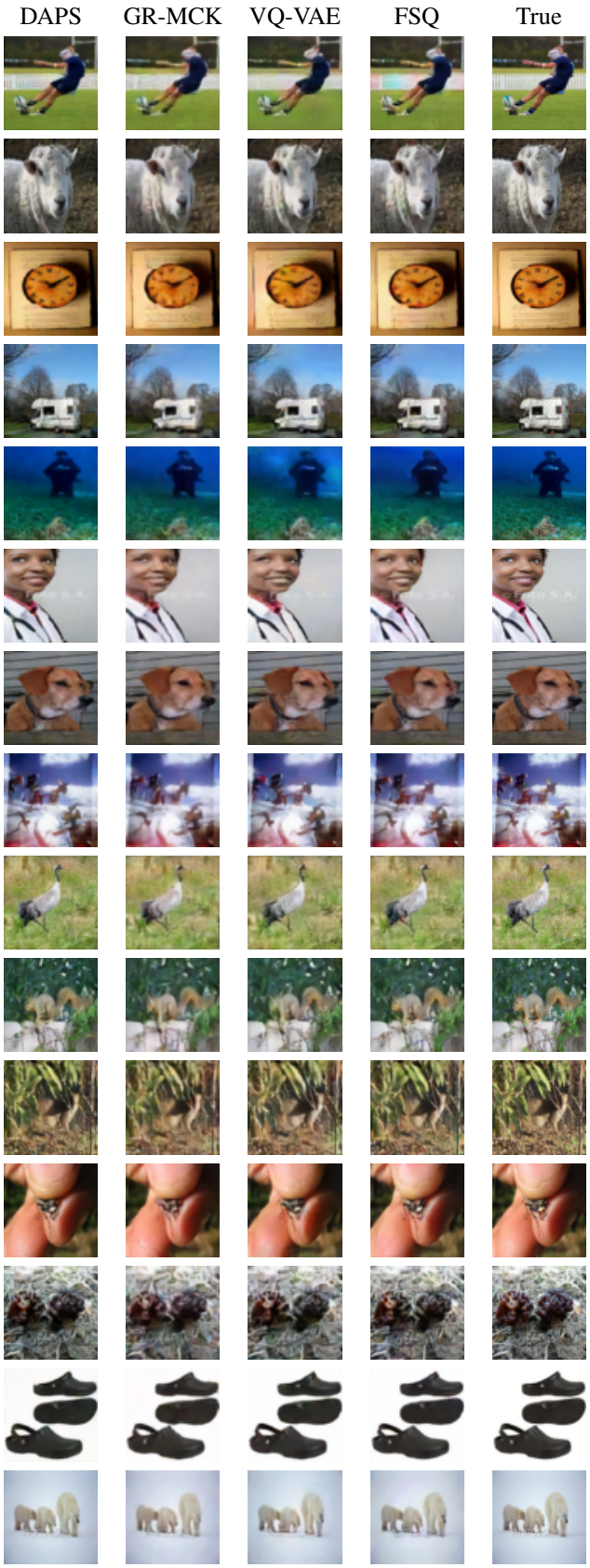


Figure 16: ImageNet 256 validation reconstructions vs. ground truth.