Just Relax It! Leveraging Relaxation for Discrete Variables Optimization

Daniil Dorin, Igor Ignashin, Nikita Kiselev, Andrey Veprikov, Ilya Stepanov, Vladislav Meshkov, Vladislav Minashkin, Ivan Papay

Intelligent Systems

MIPT 2025

Contents

List of Figures

Just Relax It! Leveraging Relaxation for Discrete Variables Optimization

***Abstract*—This report presents ”Just Relax It” (`relaxit`), a Python library designed to streamline the optimization of discrete probability distributions in neural networks. The library offers a comprehensive suite of advanced relaxation techniques compatible with PyTorch, addressing the challenges in training generative models with discrete latent variables. We demonstrate the effectiveness of our approach through Variational Autoencoder (VAE) experiments with discrete latents, showing that proper relaxation techniques enable adequate learning and reconstruction. The library implements multiple relaxation methods including Relaxed Bernoulli, Hard Concrete, Straight-Through Bernoulli, Generalized Gumbel-Softmax, Gumbel-Softmax TOP-K, REBAR, Decoupled Straight-Through Gumbel-Softmax and others, providing researchers with flexible tools for discrete variable optimization in machine learning applications. Also we implement the RELAX gradient estimator on several standard RL benchmarks to evaluate its performance in discrete action space environments.**

I. Introduction

Rapid advancement of generative models, such as Variational Autoencoders (VAEs) and Diffusion Models, has driven the development of relevant mathematical tools. Any generative model contains some source of randomness to make new objects. This randomness is represented by a probability distribution, from which random variables are sampled. Therefore, training a generative model often boils down to optimizing the parameters of this distribution.

Pioneering generative models typically work with continuous distributions like the Normal distribution. However, for some modalities, such as texts or graphs, it is more natural to use discrete distributions—Bernoulli, Categorical, etc.

Thus, we present our new Python library “Just Relax It” that combines the best techniques for relaxing discrete distributions into an easy-to-use package. And it is compatible with PyTorch.

II. Notation and conventions

To keep the notation uniform across the entire manuscript we adopt the following conventions (used everywhere below):

- x — observed data / input.
- b — binary (discrete) random variable, $b \in \{0, 1\}$.
- y — categorical discrete outcome as a one-hot vector $y \in \{e_1, \dots, e_K\}$ when needed.
- z — relaxed continuous sample (for Bernoulli: $z \in (0, 1)$; for categorical: $z \in \Delta^{K-1}$, the simplex).
- π — probability vector for categorical distributions, $\pi = (\pi_1, \dots, \pi_K)$, $\sum_k \pi_k = 1$.
- α, θ — scalar parameters for Bernoulli-like distributions (logits or probabilities depending on context).
- τ — temperature parameter controlling relaxation sharpness ($\tau > 0$).
- u (or u_i) — uniform noise drawn from $\mathrm{Uniform}(0, 1)$ (we always use u for uniform draws).
- $g = (g_1, \dots, g_K)$ — vector of independent Gumbel noise components where $g_k = -\log(-\log u_k)$ and $u_k \sim \mathrm{Uniform}(0, 1)$.
- $\mathbb{I}\{\cdot\}$ — indicator function (we use this notation consistently for indicator events).
- $H(\cdot)$ — deterministic discretization mapping (e.g. $H(z)$ is a hard threshold / argmax mapping producing a discrete variable from a relaxed z).

All inline and display formulas in the following sections adhere to these notational choices. Where an original subsection used a different local variable name (e.g. m, M, or c), we interpret it as described above (binary/categorical/relaxed sample) — and we use the symbols of this section consistently.

III. Proposed Solutions

A. Generalized Gumbel-Softmax

Overview Joo et al. [1] propose a generalized form of the Gumbel–Softmax estimator. Their main objective is to extend the reparameterization trick beyond categorical and Bernoulli variables so that a broad class of discrete distributions (e.g. Poisson, geometric, binomial, negative binomial) that have countable or large supports can be handled differentiably. To achieve this they truncate an infinite/countable support to a finite set, form categorical probabilities on that truncated support, apply a Gumbel–Softmax relaxation to obtain a soft one-hot, and finally map the relaxed one-hot back to the original discrete values via a linear transform.

Methodology Let $D(\lambda)$ be a discrete distribution parameterized by λ. After truncation we consider a finite ordered outcome vector $\mathcal{C} = (c_1, \dots, c_n)$. Define the categorical probabilities

$$\pi_k = p_\lambda(c_k), \qquad k = 1, \dots, n-1, \qquad \pi_n = 1 - \sum_{k=1}^{n-1} \pi_k,$$

so that $\sum_{k=1}^{n} \pi_k = 1$. Draw i.i.d. Gumbel noise g_k by

$$g_k = -\log(-\log(u_k)), \quad u_k \sim \mathrm{Uniform}(0, 1).$$

The Gumbel–Softmax relaxed (soft) one-hot vector w with temperature $\tau > 0$ is

$$w = \mathrm{softmax}\Big(\frac{\log \pi + g}{\tau}\Big),$$

$$w_k = \frac{\exp((\log \pi_k + g_k)/\tau)}{\sum_{j=1}^{n} \exp((\log \pi_j + g_j)/\tau)}.$$

Finally map back to the original-value sample

$$z \;=\; T(w) \;=\; \sum_{k=1}^{n} w_k\, c_k.$$

B. Relaxed Bernoulli Distribution

Overview Maddison et al. [2] and Jang et al. [3] introduced a continuous relaxation of Bernoulli and categorical variables. Their primary goal is to enable gradient-based optimization in models with binary latent variables by removing the non-differentiability of direct $\{0,1\}$ sampling. They propose a temperature-controlled reparameterization (the Relaxed Bernoulli/Concrete formulation) that maps logistic/Gumbel-style noise through a temperature-scaled sigmoid/softmax to produce a differentiable approximation in $(0,1)$ whose sharpness approaches a true discrete sample as the temperature decreases.

Methodology Formally, instead of sampling $b \sim \text{Bernoulli}(\alpha)$ where $b \in \{0,1\}$, the Relaxed Bernoulli samples continuous values $z \in (0,1)$ using the following reparameterization:

$$z = \sigma\left(\frac{\log\alpha - \log(1-\alpha) + \log u - \log(1-u)}{\tau}\right),$$

where $u \sim \text{Uniform}(0,1)$ and $\sigma(\cdot)$ is the sigmoid function. Temperature $\tau > 0$ controls relaxation sharpness.

C. Hard Concrete Distribution

Overview Louizos et al. [4] proposed the Hard Concrete distribution as a practical relaxation for learning sparse neural networks. Their aim is to combine the benefits of a differentiable relaxation during training with exact zeros at inference time so that sparsity can be realized in practice. They introduce a stretched/shifted Concrete relaxation together with a hard-sigmoid/clipping operation that produces exact 0/1 outputs at deployment while remaining differentiable during optimization, enabling effective L_0-type regularization via learned binary gates.

Methodology Given a uniform random variable $u \sim \text{Uniform}(0,1)$, samples are generated as:

$$\bar{s} = \sigma\left(\frac{\log u - \log(1-u) + \log\alpha}{\beta}\right) \cdot (\zeta - \gamma) + \gamma,$$

$$z = \max(0, \min(1, \bar{s})),$$

where $\sigma(\cdot)$ is the sigmoid function, α is the location parameter, and β is the temperature. Common values are $\gamma = -0.1$ and $\zeta = 1.1$.

D. Straight-Through Bernoulli

Overview The Straight-Through (ST) estimator — a practical technique with roots in earlier work on stochastic neurons (e.g. Bengio et al., 2013) and more formal analyses in later papers (e.g. Cheng et al., 2019) — is designed to reconcile exact discrete forward sampling with usable gradients in backpropagation. Its main objective is to obtain true binary decisions on the forward pass while providing an effective surrogate gradient for parameter updates. The method implements a hard threshold (giving $\{0,1\}$ outputs) during the forward pass and uses a differentiable proxy (or simply copies the upstream gradient) during the backward pass, offering a simple and efficient alternative to continuous relaxations for many practical tasks.

Methodology The forward pass samples uniform random variables $u_i \sim \text{Uniform}(0,1)$ and applies a hard threshold:

$$h_i = f(a_i, u_i) = \mathbb{I}\{u_i < \sigma(a_i)\},$$

where $\sigma(a_i)$ is the sigmoid-transformed logit parameter. The indicator function $\mathbb{I}\{\cdot\}$ produces exact binary outputs $\{0,1\}$. During the backward pass, the gradient of the loss L with respect to the logit a_i is typically approximated by a straight-through surrogate, for example:

$$\frac{\partial L}{\partial a_i} \approx \frac{\partial L}{\partial h_i} \cdot \sigma'(a_i),$$

or simply copying $\partial L/\partial h_i$ into the slot for $\partial L/\partial a_i$ depending on the chosen ST variant.

E. Stochastic Times Smooth

Overview Bengio et al. [5] introduced the Stochastic Times Smooth (STS) idea as a more principled way to train stochastic binary neurons. The core objective is to keep exact binary stochasticity in the forward pass while enabling gradient flow in a principled manner during backpropagation, avoiding purely heuristic gradient copying. STS decomposes the neuron into a stochastic binary component and a smooth scaling factor so that the expected behaviour of the stochastic unit is approximated in a differentiable way, improving optimization stability relative to naïve straight-through methods.

Methodology The STS unit generates outputs through a two-step process that separates stochastic sampling from differentiable scaling:

$$\begin{aligned} p_i &= \sigma(a_i), \\ b_i &\sim \text{Bernoulli}(\sqrt{p_i}), \\ h_i &= b_i \cdot \sqrt{p_i}, \end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid function, a_i is the pre-activation logit, b_i is a Bernoulli random variable with probability $\sqrt{p_i}$, and h_i is the final (scaled) output.

F. Correlated Relaxed Bernoulli

Overview Lee et al. [6] (and related recent works on correlated relaxed gates) proposed a correlated extension of the relaxed Bernoulli aimed at modeling dependencies between binary decisions. The main motivation is that in many applications the binary variables are not independent (feature selection, structured gates), and naive independent relaxations ignore that structure. They propose to generate correlated relaxed gate vectors by coupling marginal relaxed Bernoulli samples through a Gaussian copula (i.e. sample correlated Gaussian noise, transform to uniforms, then apply the logistic/sigmoid reparameterization), thereby producing a differentiable, multivariate relaxation that captures dependencies between dimensions.

Methodology The key innovation lies in reparameterizing correlated binary variables through a shared noise source. The relaxed gate vector $\tilde{z}_k$ is generated using the reparameterization trick:

$$\tilde{z}_k = \sigma\left(\frac{1}{\tau}\left(\log \pi_k - \log(1-\pi_k) + \log U_k - \log(1-U_k)\right)\right),$$

where $U_k \sim \mathrm{Uniform}(0,1)$ is a uniform random variable obtained from a Gaussian copula (to induce correlations), π_k denotes the success probability for dimension k, and τ is a temperature hyperparameter controlling relaxation sharpness. The logarithmic ratio $\log U_k - \log(1-U_k)$ transforms uniform noise into a logistic-like perturbation and enables differentiable correlated sampling.

G. Invertible Gaussian

Overview Potapczynski et al. [7] revisit the Gumbel–Softmax by proposing a flexible family of reparameterizations based on Gaussian noise followed by invertible smooth transforms. Their objective is to provide a modular and more flexible alternative to Gumbel–Softmax that allows closed-form KL computations and the insertion of invertible transforms (flows) before mapping to the simplex. Concretely, they sample Gaussian noise, apply a location–scale transform, and then use an invertible mapping to the simplex to obtain relaxed categorical-like samples with desirable analytic properties.

Methodology Sample Gaussian noise $\varepsilon \sim \mathcal{N}(0, I_{K-1})$. Apply location-scale transformation:

$$y = \mu + \mathrm{diag}(\sigma)\varepsilon.$$

Map to simplex via temperature-controlled function

$$\tilde{z} = g(y, \tau),$$

where $\mu \in \mathbb{R}^{K-1}$ and $\sigma \in (0,\infty)^{K-1}$ are parameters, $\tau > 0$ is a temperature hyperparameter, and $g(\cdot, \tau)$ is an invertible smooth function mapping to the simplex $S^{(K-1)} = \{z \in \mathbb{R}^K : z_k > 0,\ \sum_{k=1}^K z_k = 1\}$.

H. Gumbel-Softmax TOP-K

Overview Kool et al. [8] introduce the Gumbel-Top-k trick to sample k distinct elements without replacement from a categorical distribution. Their goal is to generalize the single-sample Gumbel–Max trick so that multiple items can be drawn in one pass while preserving correct sampling probabilities. They show that adding independent Gumbel noise to log-probabilities and taking the top-k perturbed values yields an exact sample without replacement, and they provide efficient algorithms (e.g. stochastic beam search) to apply this idea at scale.

Methodology Given a categorical distribution over n elements with probabilities p_i and log-probabilities $\phi_i = \log p_i$. For each element $i \in \{1, \ldots, n\}$, add independent Gumbel noise to its log-probability:

$$G_{\phi_i} = \phi_i + G_i, \quad \text{where } G_i \sim \mathrm{Gumbel}(0)$$

The Gumbel(0) distribution can be sampled via $G_i = -\log(-\log(U_i))$ with $U_i \sim \mathrm{Uniform}(0,1)$. Identify the indices of the k largest perturbed log-probabilities:

$$I_1^*, I_2^*, \ldots, I_k^* = \mathrm{argtopk}_i\, G_{\phi_i},$$

where argtopk returns the indices of the k largest values in descending order. The ordered sequence $(I_1^*, I_2^*, \ldots, I_k^*)$ constitutes a sample of k elements drawn without replacement from the categorical distribution. The joint probability of a specific ordered sample $(i_1, i_2, \ldots, i_k)$ is given by:

$$P(I_1^* = i_1, \ldots, I_k^* = i_k) = \prod_{j=1}^{k} \frac{\exp(\phi_{i_j})}{\sum_{l \in N_j} \exp(\phi_l)},$$

where $N_j = \{1, \ldots, n\} \setminus \{i_1, \ldots, i_{j-1}\}$ is the set of remaining elements after the first $j-1$ selections. This corresponds exactly to sequential sampling without replacement, where at each step the probability of selecting a remaining element is proportional to its original probability p_i.

I. Closed-form Laplace Bridge

Overview Hobbhahn et al. [9] revisit the Laplace Bridge idea to obtain a closed-form Dirichlet approximation of the softmax-of-Gaussian predictive distribution (see Hobbhahn et al., 2020/2022). Their objective is to avoid expensive sampling from a Gaussian over logits and instead provide an analytic mapping from a Gaussian in logit space to a Dirichlet on the probability simplex, enabling fast uncertainty estimation for classification outputs. The proposal yields explicit formulae that relate Gaussian parameters (μ, Σ) to Dirichlet parameters α, providing a computationally cheap approximation for downstream use in Bayesian deep learning.

Methodology The Laplace Bridge (LB) method provides an analytic mapping between the parameters of a Gaussian distribution over logits and the parameters of a Dirichlet distribution over probabilities in the simplex.

The Dirichlet distribution over the probability simplex $\pi \in \mathbb{R}^K$ with parameters $\alpha \in \mathbb{R}_+^K$ has the density:

$$\mathrm{Dir}(\pi \mid \alpha) := \frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1},$$

where $\Gamma(\cdot)$ is the gamma function.

Consider the variable π obtained by applying the softmax to a vector $z \in \mathbb{R}^K$:

$$\pi_k(z) := \frac{\exp(z_k)}{\sum_{l=1}^K \exp(z_l)}, \quad k = 1, \ldots, K.$$

Then the density of the Dirichlet distribution, expressed in terms of z, must be multiplied by the Jacobian of the transformation:

$$\left|\det \frac{\partial \pi}{\partial z}\right| = \prod_{k=1}^{K} \pi_k(z),$$

which leads to the modified density:

$$\mathrm{Dir}_z(\pi(z) \mid \alpha) := \frac{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \pi_k(z)^{\alpha_k}.$$

The Laplace Bridge establishes an analytic correspondence between the parameters α of the Dirichlet distribution and the parameters (μ, Σ) of the Gaussian approximation for z:

$$\mu_k = \log \alpha_k - \frac{1}{K} \sum_{l=1}^{K} \log \alpha_l,$$

$$\Sigma_{kl} = \delta_{kl} \frac{1}{\alpha_k} - \frac{1}{K} \left(\frac{1}{\alpha_k} + \frac{1}{\alpha_l} - \frac{1}{K} \sum_{u=1}^{K} \frac{1}{\alpha_u} \right),$$

where δ_{kl} is the Kronecker delta.

The inverse transformation (pseudo-inverse mapping) allows obtaining the Dirichlet parameters from the Gaussian approximation:

$$\alpha_k = \frac{1}{\Sigma_{kk}} \left(1 - \frac{2}{K} + \frac{2}{K^2} \sum_{l=1}^{K} e^{\mu_k - \mu_l} \right)^{-1}.$$

This equation ignores the off-diagonal elements of Σ.

J. RELAX (REinforcement Learning with EXpected gradients)

Overview Grathwohl et al. [10] introduce RELAX, a general framework that learns a neural control variate to produce low-variance, unbiased gradient estimators for functions of discrete random variables. The primary aim is to combine the low variance of reparameterization-based estimators with the unbiasedness of score-function (REINFORCE) estimators, while avoiding manual control-variates. RELAX achieves this by introducing a learned control variate c_ϕ evaluated on both unconditional and conditional relaxations of the discrete variable and jointly optimizing the variate parameters to minimize gradient variance, producing an unbiased but much lower-variance estimator.

Methodology RELAX aims to estimate $\nabla_\theta \mathbb{E}_{p(b|\theta)}[f(b)]$, where b is a discrete random variable. The method introduces a relaxed continuous variable $z \sim p(z|\theta)$ and a deterministic mapping $H(z) = b$. This leads to the DLAX estimator:

$$\hat{g}_{\mathrm{DLAX}} = f(b) \nabla_\theta \log p(b|\theta) - c_\phi(z) \nabla_\theta \log p(z|\theta) + \nabla_\theta c_\phi(z).$$

However, the DLAX estimator suffers from poor correlation between the log-derivative terms. To address this, RELAX introduces a conditional distribution $\tilde{z} \sim p(z|b,\theta)$, allowing the control variate to be evaluated at both unconditional ($z \sim p(z|\theta)$) and conditional ($\tilde{z} \sim p(z|b,\theta)$) relaxations.

The RELAX estimator is then formulated as:

$$\hat{g}_{\mathrm{RELAX}} = [f(b) - c_\phi(\tilde{z})] \nabla_\theta \log p(b|\theta) + \nabla_\theta c_\phi(z) - \nabla_\theta c_\phi(\tilde{z}),$$

where $b = H(z)$, $z \sim p(z|\theta)$, and $\tilde{z} \sim p(z|b,\theta)$.

The parameters ϕ of the control variate are optimized by minimizing the variance of the gradient estimator. A single-sample unbiased estimate is given by:

$$\hat{g}_\phi = \frac{\partial \hat{g}_\theta^2}{\partial \phi}.$$

This enables joint optimization of both the model parameters θ and the control variate parameters ϕ during training, making RELAX a powerful and flexible tool for gradient estimation in models with discrete variables.

K. Decoupled Straight-Through Gumbel-Softmax

Overview Shah et al. [11] propose a novel decoupling of forward and backward pass temperatures for Straight-Through Gumbel-Softmax (ST-GS) to address the fundamental trade-off between discrete sampling and gradient quality. Standard ST-GS uses a single temperature τ for both operations, limiting the ability to optimize discretization sharpness independently from gradient smoothness. The method introduces separate temperature parameters to enable sharp discrete forward passes while maintaining smooth, low-variance backward gradients, achieving superior performance particularly for tasks requiring precise categorical decisions such as neural architecture search and hard attention mechanisms.

Methodology The decoupled ST-GS uses separate temperature parameters: τ^f for the forward discrete sampling and τ^b for backward gradient approximation. The forward pass performs discrete sampling via argmax:

$$z = \text{one-hot}\left(\text{argmax}\left(\text{softmax}\left(\frac{l+g}{\tau^f}\right)\right)\right),$$

where l represents input logits and $g \sim \mathrm{Gumbel}(0,1)$.

The backward pass computes smooth gradients through a differentiable relaxation:

$$\frac{\partial \mathcal{L}}{\partial l} \approx \frac{\partial \mathcal{L}}{\partial z} \frac{\partial \hat{z}^b}{\partial l}, \quad \hat{z}^b = \text{softmax}\left(\frac{l+g}{\tau^b}\right).$$

The optimal configuration uses $\tau^f < \tau^b$, preserving discrete structure in activations while ensuring stable optimization dynamics.

L. REBAR Relaxation

Overview The REBAR [12] method combines the low-variance but biased Concrete relaxation with the unbiased but high-variance REINFORCE estimator through a novel control variate.

Methodology For binary random variables $b \sim \mathrm{Bernoulli}(\theta)$, we want to estimate:

$$\nabla_\theta \mathbb{E}_{p(b)}[f(b)] = \mathbb{E}_{p(b)}\left[f(b) \nabla_\theta \log p(b)\right].$$

Parameterize $b = H(z)$ where z are Logistic random variables:

$$z = g(u,\theta) = \log \frac{\theta}{1-\theta} + \log \frac{u}{1-u}, \quad u \sim \mathrm{Uniform}(0,1).$$

Replace hard threshold $H(z)$ with sigmoid relaxation $\sigma_\lambda(z) = \sigma(z/\lambda)$.

The REBAR gradient estimator is:

$$\nabla_\theta \mathbb{E}_{p(b)}[f(b)] = \mathbb{E}_{p(u,v)} \Big[[f(H(z)) - \eta f(\sigma_\lambda(\tilde{z}))] \nabla_\theta \log p(b)\big|_{b=H(z)} + \eta \nabla_\theta f(\sigma_\lambda(z)) - \eta \nabla_\theta f(\sigma_\lambda(\tilde{z}))\Big],$$

where:

- $z = g(u, \theta)$,
- $\tilde{z} = \tilde{g}(v, H(z), \theta)$ is the reparameterization of $p(z|b)$,
- η is a scaling parameter optimized to minimize variance.

The REBAR estimator remains unbiased for any temperature parameter $\lambda > 0$, ensuring mathematical correctness regardless of the relaxation tightness. REBAR exhibits an important theoretical connection to MuProp: as $\lambda \to \infty$, the REBAR estimator converges to the MuProp estimator without its linear term, providing a smooth interpolation between these approaches.

IV. Research Methodology

Our research methodology involved comprehensive implementation and experimental validation of each relaxation technique. We followed a systematic approach:

- **Library Design**: We built upon PyTorch's distribution framework, ensuring compatibility with existing deep learning workflows
- **Algorithm Implementation**: Each relaxation method was implemented as a custom distribution class inheriting from TorchDistribution
- **Experimental Validation**: We conducted controlled experiments on standard dataset (MNIST) to evaluate each method's performance on VAE

The implementation focused on providing consistent interfaces while maintaining the mathematical correctness of each relaxation technique. All distributions implement essential methods including `rsample()` for reparameterized sampling and `log_prob()` for likelihood computation.

In our library, you can also find an implementation of the Laplace bridge between the Dirichlet and Logistic–Normal distributions.

To evaluate RELAX alongside REINFORCE and A2C, a unified benchmarking pipeline was implemented in PyTorch using Gym environments. The RELAX agent is implemented as a policy gradient method for discrete action spaces that combines a feed-forward actor network with a control-variate critic conditioned on Gumbel-Softmax relaxations of the discrete actions. The training loop for RELAX collects trajectories with both relaxed and conditional Gumbel samples, computes low-variance, unbiased gradient estimates using the RELAX estimator, and alternates critic updates (to minimize gradient variance) with custom actor updates based on the accumulated RELAX gradients. This implementation is used as the final branch of the methodology to generate the learning curves summarized in Figure 3.

V. Analysis and Interpretation

Our experimental results demonstrate the effectiveness of relaxation techniques for discrete variable optimization. Implementations with relaxations on VAE are given in the demo. For example, the VAE with discrete latents trained using Correlated Relaxed Bernoulli achieved promising results on MNIST dataset.

The reconstruction results (Fig. 1) show that the model successfully learned to encode and decode digit images, while the sampling results (Fig. 2) demonstrate that the latent space captured meaningful variations in the data. This suggests that the reparameterization is occurring correctly and the gradients are propagating effectively through the discrete latent variables.

The evidence indicates that relaxation techniques can successfully bridge the gap between discrete distributions and gradient-based optimization, though the choice of method depends on the specific application requirements and constraints.

Figure 3 compares the learning curves of REINFORCE, A2C, and RELAX on the Cart-pole-v1 (left) and Taxi-v3 (right) benchmarks. On Cart-pole, RELAX exhibits stable and nearly linear reward growth, outperforming REINFORCE and approaching the performance of A2C while avoiding noticeable instability or collapse.

On Taxi-v3, RELAX clearly dominates both baselines in Figure 3, starting from a higher reward and improving at a faster rate throughout training. Its smoother and steeper learning curve indicates that RELAX handles the more complex, discrete structure of Taxi-v3 more effectively, converging to higher-quality policies with fewer plateaus than REINFORCE and A2C.

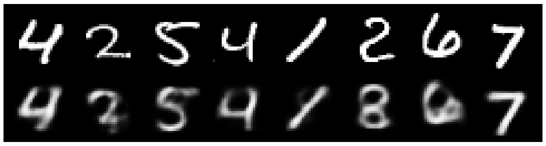

Fig. 1. VAE with Correlated Relaxed Bernoulli latents. Reconstruction.

Experiments (Fig. 4, 5, 6) related to the Laplace bridge between the Logistic-normal and the dirichlet distribution are also demonstrated.

VI. Conclusions and Recommendations

In summary, Just Relax It is a powerful tool for researchers and practitioners working with discrete variables in neural networks. By offering a comprehensive set of relaxation techniques, our library aims to make the optimization process more efficient and accessible.

The library successfully addresses the challenge of discrete variable optimization and enables effective training of generative models with discrete latent spaces. We encourage further research into hybrid methods and automatic selection of relaxation techniques based on problem characteristics.

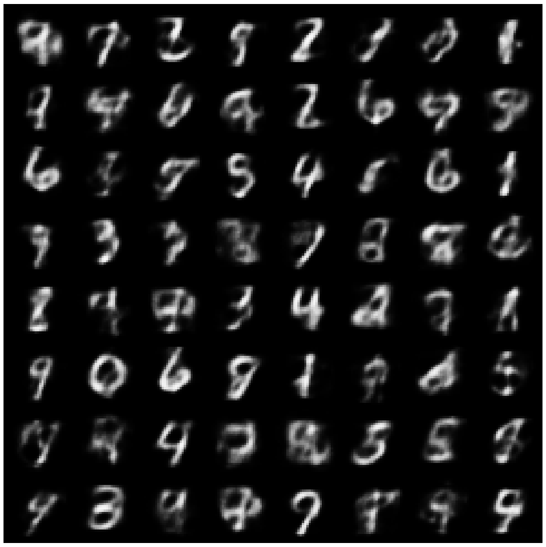

Fig. 2. VAE with Correlated Relaxed Bernoulli latents. Sampling.

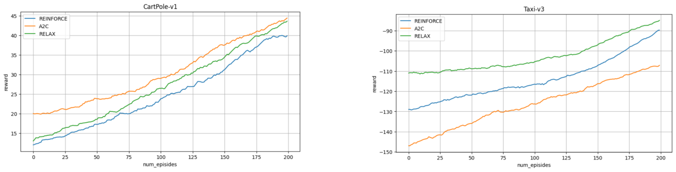

Fig. 3. Comparison of A2C, REINFORCE, and RELAX algorithms on the Cart-pole (left) and Taxi-v3 (right) benchmarks. A2C employs an actor-critic architecture for reduced gradient variance, REINFORCE uses a pure policy gradient approach, and RELAX leverages relaxation techniques for improved convergence and stability in reinforcement learning tasks

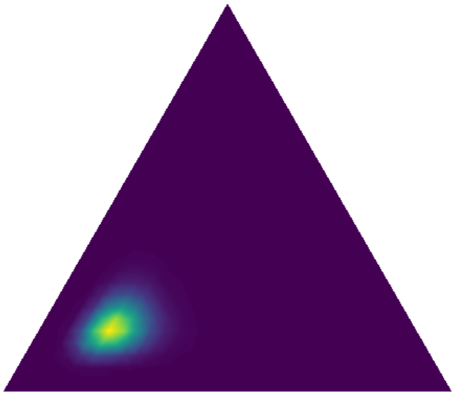

Fig. 4. Logistic-Normal (approximation to Dirichlet)

Appendix A
Demo Experiments

Our demo code is available at GitHub repository. The experiments are divided into two groups:

- **Laplace Bridge Demonstration**: Shows the closed-form approximation between Dirichlet and Logistic-Normal distributions

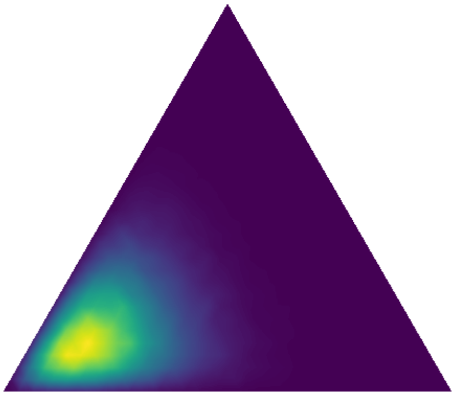

Fig. 5. Dirichlet (with random parameters)

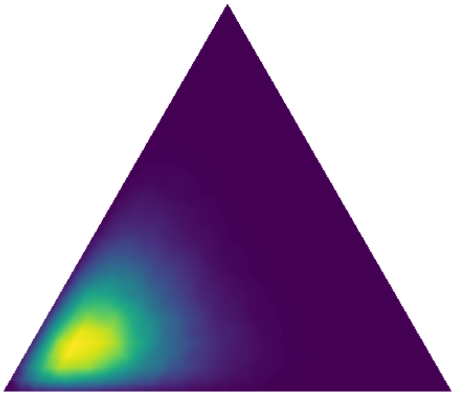

Fig. 6. Dirichlet (approximation to Logistic-Normal)

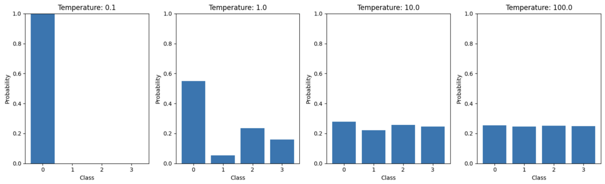

Fig. 7. Effect of Temperature on Distribution Sharpness for Decoupled ST-GS

- **VAE with Discrete Latents**: Trains Variational Autoencoders using all relaxation methods on MNIST dataset
- **RELAX** on Reinforcement Learning Benchmarks: We implement the RELAX gradient estimator on several standard RL benchmarks to evaluate its performance in discrete action space environments. The experiments include classic control tasks: CartPole, Acrobot and environments from OpenAI Gym, where RELAX is compared against REINFORCE and A2C.

The Correlated Relaxed Bernoulli demonstration uses parameters: $\pi = [[0.2, 0.4, 0.4], [0.3, 0.5, 0.2]]$, correlation matrix R with values $[1.0, 0.5, 0.3; 0.5, 1.0, 0.7; 0.3, 0.7, 1.0]$, and temperature $\tau = 0.1$.

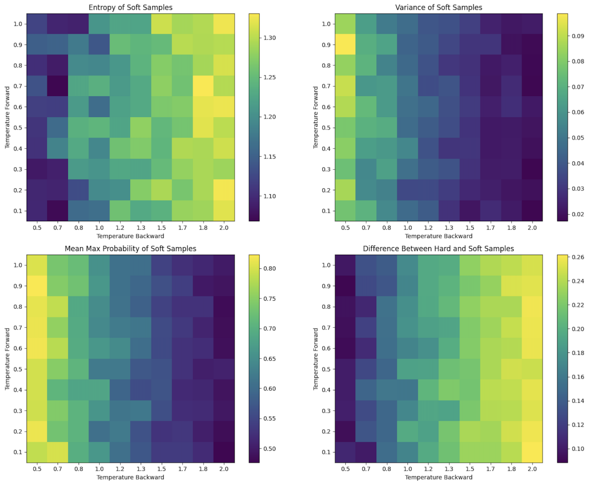

Fig. 8. Effect of Temperature Combinations on Sampling for Decoupled ST-GS

References

[1] W. Joo, D. Kim, S. Shin, and I.-C. Moon, "Generalized gumbel-softmax gradient estimator for generic discrete random variables," *arXiv preprint arXiv:2003.01847*, 2020.

[2] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[3] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[4] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.

[5] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[6] C. Lee, F. Imrie, and M. van der Schaar, "Self-supervision enhanced feature selection with correlated gates," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=oDFvtxzPOx

[7] A. Potapczynski, G. Loaiza-Ganem, and J. P. Cunningham, "Invertible gaussian reparameterization: Revisiting the gumbel-softmax," *arXiv preprint arXiv:1912.09588*, 2019.

[8] W. Kool, H. Van Hoof, and M. Welling, "Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement," in *International conference on machine learning*. PMLR, 2019, pp. 3499–3508.

[9] M. Hobbhahn, A. Kristiadi, and P. Hennig, "Fast predictive uncertainty for classification with bayesian deep networks," in *Uncertainty in artificial intelligence*. PMLR, 2022, pp. 822–832.

[10] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud, "Backpropagation through the void: Optimizing control variates for black-box gradient estimation," *arXiv preprint arXiv:1711.00123*, 2017.

[11] R. Shah, M. Yan, M. C. Mozer, and D. Liu, "Improving discrete optimisation via decoupled straight-through gumbel-softmax," *arXiv preprint arXiv:2410.13331*, 2024.

[12] G. Tucker, A. Mnih, C. J. Maddison, J. Lawson, and J. Sohl-Dickstein, "Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models," *Advances in Neural Information Processing Systems*, vol. 30, 2017.