Resampled Priors for Variational Autoencoders

Matthias Bauer[1]
Max Planck Institute for Intelligent Systems
University of Cambridge

Andriy Mnih
DeepMind

Abstract

We propose Learned Accept/Reject Sampling (Lars), a method for constructing richer priors using rejection sampling with a learned acceptance function. This work is motivated by recent analyses of the VAE objective, which pointed out that commonly used simple priors can lead to underfitting. As the distribution induced by Lars involves an intractable normalizing constant, we show how to estimate it and its gradients efficiently. We demonstrate that Lars priors improve VAE performance on several standard datasets both when they are learned jointly with the rest of the model and when they are fitted to a pretrained model. Finally, we show that Lars can be combined with existing methods for defining flexible priors for an additional boost in performance.

1 Introduction

Variational Autoencoders (VAEs) [17, 25] are powerful and widely used probabilistic generative models. In their original formulation, both the prior and the variational posterior are parameterized by factorized Gaussian distributions. Many approaches have proposed using more expressive distributions to overcome these limiting modelling choices; most of these focus on flexible variational posteriors [e.g. 15, 24, 27, 31]. More recently, both the role of the prior and its mismatch to the *aggregate* variational posterior have been investigated in detail [12, 26]. It has been shown that the prior minimizing the KL term in the ELBO is given by the corresponding aggregate posterior [30]; however, this choice usually leads to overfitting as this kind of non-parametric prior essentially memorizes the training set. The VampPrior [30] models the aggregate posterior explicitly using a mixture of posteriors of learned virtual observations with a fixed number of components. Other ways of specifying expressive priors include flow-based [4] and autoregressive models [10], which provide different tradeoffs between expressivity and efficiency.

In this work, we present Learned Accept/Reject Sampling, an alternative way to address the mismatch between the prior and the aggregate posterior in VAEs. The proposed Lars prior is the result of applying a rejection sampler with a learned acceptance function to the original prior, which now serves as the proposal distribution. The acceptance function, typically parameterized using a neural network, can be learned either jointly with the rest of the model by maximizing the ELBO, or separately by maximizing the likelihood of samples from the aggregate posterior of a trained model. Our approach is orthogonal to most existing approaches for making priors more expressive and can be easily combined with them by using them as proposals for the sampler. The price we pay for richer priors is iterative sampling, which can require rejecting a large number of proposals before accepting one. To improve the acceptance rate and, thus, the sampling efficiency of the sampler, we introduce a truncated version that always accepts the next proposal if some predetermined number of samples have been rejected in a row.

Lars is neither limited to variational inference nor to continuous variables. In fact, it can be thought of as a generic method for transforming a tractable base distribution into a more expressive one by modulating its density while keeping learning or exact sampling tractable.

Our main contributions are as follows:

- We introduce Learned Accept/Reject Sampling, a method for transforming a tractable base distribution into a better approximation to a target density, based only on samples from the latter.
- We develop a truncated version of Lars which enables a principled trade-off between approximation

[1]Work done during an internship at DeepMind.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

accuracy and sampling efficiency.

- We apply LARS to VAEs, replacing the standard Normal priors with LARS priors. This consistently yields improved ELBOs, with further gains achieved using expressive flow-based proposals.
- We demonstrate the versatility of our method by applying it to the high-dimensional discrete output distribution of a VAE.

2 Variational Autoencoders

A VAE [17, 25] models the distribution of observations $\mathbf{x}$ using a stochastic latent vector $\mathbf{z}$, by specifying its prior $p(\mathbf{z})$ along with a likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ that connects it with the observation. As computing the marginal likelihood of an observation $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})d\mathbf{z}$ and, thus, maximum likelihood learning are intractable in VAEs, they are trained by maximizing the variational *evidence lower bound* (ELBO) on the marginal log-likelihood:

$$\mathbb{E}_{q(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\lambda(\mathbf{z}))\right] \quad (1)$$

where $q(\mathbf{x})$ is the empirical distribution of the training set and $q_\phi(\mathbf{z}|\mathbf{x})$ is the *variational posterior*. The neural networks used to parameterize $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ are referred to as the *encoder* and *decoder*, respectively. In the rest of the paper we omit the parameter subscripts to reduce notational clutter. The gradients of the ELBO w.r.t. all parameters can be estimated efficiently using the reparameterization trick [17, 25].

The first term in Eq. (1), the (negative) reconstruction error, measures how well a sample $\mathbf{x}$ can be reconstructed by the decoder from its latent space representation produced by the encoder. The second term, the Kullback-Leibler (KL) divergence between the variational posterior and the prior, acts as a regularizer that limits the amount of information passing through the latents and ensures that the model can also generate the data rather than just reconstruct it. We can rewrite the KL in terms of the *marginal* or *aggregate (variational) posterior* $q(\mathbf{z}) = \mathbb{E}_{q(\mathbf{x})}[q(\mathbf{z}|\mathbf{x})]$ [12]:

$$\mathbb{E}_{q(\mathbf{x})} \mathrm{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z})) + I(\mathbf{x}; \mathbf{z}), \quad (2)$$

where the second term is the mutual information between the latent vector and the training example. This formulation makes it clear that maximizing the ELBO corresponds to minimizing the KL between the aggregate posterior and the prior. Despite this, there usually is a mismatch between the two distributions in trained models [12, 26]. This phenomenon is sometimes described as "holes in the aggregate posterior", referring to the regions of the latent space that have high density under the prior but very low density under the aggregate posterior, meaning that they are almost never encountered during training. Samples from these regions are typically decoded to observations that do not lie on the data manifold [26].

Our goal is to make the prior $p(\mathbf{z})$ sufficiently expressive to make it substantially closer to $q(\mathbf{z})$ compared to the original prior, without making the two identical, as that can result in overfitting. In the following we introduce a simple and effective way of achieving this by learning to reject the samples from the proposal that have low density under the aggregate posterior.

3 Learned Accept/Reject Sampling

In this section we present our main contribution: *Learned Accept/Reject Sampling (*LARS*), a practical method to approximate a target density q from which we can sample but whose density is either too expensive to evaluate or unknown.* It works by reweighting a simpler proposal distribution π using a learned acceptance function a as visualized in Fig. 1. The aggregate posterior of a VAE, which is a very large mixture, is one example of such a target distribution. For simplicity, we only deal with continuous variables in this section; however, our approach is equally applicable to discrete random variables as we explain in Section 5.3.

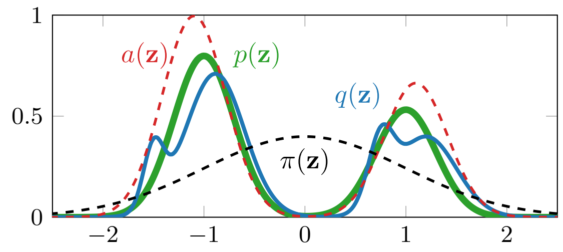

Figure 1: LARS approximates a target density $q(\mathbf{z})$ (—) by a resampled density $p(\mathbf{z})$ (—) obtained by multiplying a proposal $\pi(\mathbf{z})$ (- - -) with a learned acceptance function $a(\mathbf{z}) \in [0, 1]$ (- - -) and normalizing.

Approximating a target by modulating a proposal. Our goal is to approximate a complex target distribution $q(\mathbf{z})$ that we can sample from but whose density we are unable to evaluate. We start with an easy-to-sample-from base distribution $\pi(\mathbf{z})$ with a tractable density (with the same support as q), which might have learnable parameters but is insufficiently expressive to approximate $q(\mathbf{z})$ well. We then transform $\pi(\mathbf{z})$ into a more expressive distribution $p_\infty(\mathbf{z})$ by multiplying it by an *acceptance function* $a_\lambda(\mathbf{z}) \in [0, 1]$ and renormalizing to obtain the density

$$p_\infty(\mathbf{z}) = \frac{\pi(\mathbf{z})a_\lambda(\mathbf{z})}{Z}; \qquad Z = \int \pi(\mathbf{z})a_\lambda(\mathbf{z})d\mathbf{z}, \quad (3)$$

where Z is the normalization constant and λ are the learnable parameters of $a_\lambda(\mathbf{z})$; we drop the subscript λ in the following. Similar to other approaches for specifying expressive densities, the parameters of $a(\mathbf{z})$ can

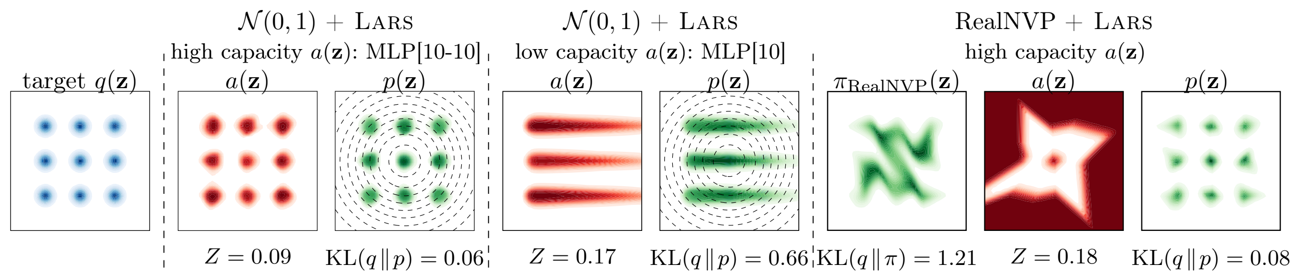

Figure 2: Learned acceptance functions $a(\mathbf{z})$ (red) that approximate a fixed target $q(\mathbf{z})$ (blue) by reweighting a $\mathcal{N}(0,1)$ (- - -) or a RealNVP (green) proposal to obtain an approximate density $p(\mathbf{z})$ (green). $\mathrm{KL}\big(q\,\big\|\,\pi_{\mathcal{N}(0,1)}\big) = 1.8$.

be learned end-to-end by optimizing the natural objective for the task at hand. p_∞ is also the distribution we obtain by using rejection sampling [33] with $\pi(\mathbf{z})$ as the proposal and $a(\mathbf{z})$ as the acceptance probability function. Thus, we can sample from p_∞ as follows: draw a candidate sample $\mathbf{z}$ from the proposal π and accept it with probability $a(\mathbf{z})$; if accepted, return $\mathbf{z}$ as the sample; otherwise repeat the process. This means we can think of $a(\mathbf{z})$ as a filtering function which — by rejecting different fractions of samples at different locations — can transform $\pi(\mathbf{z})$ into a distribution closer to $q(\mathbf{z})$. In fact, as we show in Section 3.2, if the capacity of $a(\mathbf{z})$ is unlimited, we can obtain $p_\infty(\mathbf{z})$ that matches the target distribution perfectly.

The efficiency of the sampler is closely related to the normalizing constant Z, which can be interpreted as the average probability of accepting a candidate sample. Thus, on average, we will consider $1/Z$ candidates before accepting one. Interestingly, the same distribution can be induced by rejection samplers of very different efficiency, since scaling $a(\mathbf{z})$ by a constant (while keeping a in the $[0,1]$ range) has no effect on the resulting $p_\infty(\mathbf{z})$ because Z gets scaled accordingly. This means that learning an efficient sampler might require explicitly encouraging a to be as large as possible.

Truncated rejection sampling. We now describe a simple and effective way to guarantee a minimum sampling efficiency that uses a modified density p_T instead of p_∞. We derive it from a *truncated sampling scheme* (see Appendix A.2 for the derivation): instead of allowing an arbitrary number of candidate samples to be rejected before accepting one, we cap this number and always accept the T^{th} sample if the preceding $T-1$ samples have been rejected. The corresponding density is a mixture of $p_\infty(\mathbf{z})$ and the proposal $\pi(\mathbf{z})$, with the mixing weights determined by the average rejection probability $(1-Z)$ and the truncation parameter T:

$$p_T(\mathbf{z}) = (1-\alpha_T)\frac{a(\mathbf{z})\pi(\mathbf{z})}{Z} + \alpha_T\pi(\mathbf{z}), \qquad (4)$$

where $\alpha_T = (1-Z)^{T-1}$. Note that for $T=1$ we recover the proposal density π, while letting $T\to\infty$ yields the density for untruncated sampling (Eq. (3)).

While the truncated sampling density will in general be less expressive due to smoothing by interpolation with π, its acceptance rate is guaranteed to be at least $1/T$, with the expected number of resampling steps being $(1-(1-Z)^T)/Z \le T$; see Appendix A.2 for details. Thus we can trade off approximation accuracy against sampling efficiency by varying T. Using the truncated sampling density also has the benefit of providing a natural scale for a in an interpretable and principled manner as, unlike $p_\infty(\mathbf{z})$, $p_T(\mathbf{z})$ is not invariant under scaling a by $c \in (0,1]$.

Estimating Z. Estimating the normalizing constant Z is the main technical challenge of our method. An estimate of Z is needed both for updating the parameters of $p_T(\mathbf{z})$ and for evaluating trained models. For reliable model evaluation, we use the Monte Carlo estimate

$$Z \approx \frac{1}{S}\sum_{s=1}^{S} a(\mathbf{z}_s) \qquad \mathbf{z}_s \sim \pi(\mathbf{z}), \qquad (5)$$

which we compute only once, using a very large S when the model is fully trained. For training, we found it sufficient to estimate Z once per batch using a modest number of samples. In the forward pass, we use a version of Z that is smoothed with exponential averaging, whereas the gradients in the backward pass are computed on samples from the current batch only. We also include a reweighted sample from the target in the estimator to decrease the variance of the gradients (similar to [1]). See Appendix A.4 for a detailed explanation.

3.1 Illustrative examples in 2D

Before applying Lars to VAEs, we investigate its ability to fit a mixture of Gaussians (MoG) target distribution in 2D, see Fig. 2 *(left)*. We use a standard Normal distribution as the proposal and train an acceptance function parameterized by a small MLP, to reweight π to q by maximizing the supervised objective

$$\mathbb{E}_{q(\mathbf{z})}\left[\log p_\infty(\mathbf{z})\right] = \mathbb{E}_{q(\mathbf{z})}\left[\log\frac{\pi(\mathbf{z})a(\mathbf{z})}{Z}\right], \qquad (6)$$

which amounts to maximum likelihood learning.

We find that a small network for $a(\mathbf{z})$ (an MLP with two layers of 10 units) is sufficient to separate out the individual modes almost perfectly (Fig. 2 (*left*)). However, an even simpler network (one layer with 10 units) already can learn to cut away the unnecessary tails of the proposal and leads to a better fit than the proposal (Fig. 2 (*middle*)). While samples from the higher-capacity model are closer to the target (lower KL to the target), the sampling efficiency is lower (smaller Z).

As a second example, we consider the same target distribution but use a RealNVP [6] proposal which is trained jointly with the acceptance function, see Fig. 2 (right). Trained alone, the RealNVP warps the standard Normal distribution into a multi-modal distribution, but fails to isolate all the modes (Fig. C.4 in the Appendix, also shown by Huang et al. [13]). On the other hand, the model obtained by using Lars with a RealNVP proposal manages to isolate all the modes. It also achieves higher average accept probability than Lars with a standard Normal proposal, as the RealNVP proposal approximates the target better, see Fig. 2 (right).

3.2 Relationship to rejection sampling

Here we briefly review classical rejection sampling (RS) [33] and relate it to Lars. RS provides a way to sample from a target distribution with a known density $q(\mathbf{z})$ which is hard to sample from directly but is easy to evaluate point-wise. Note that this is *exactly the opposite* of our setting, in which $q(\mathbf{z})$ is easy to sample from but infeasible to evaluate. By drawing samples from a tractable proposal distribution $\pi(\mathbf{z})$ and accepting them with probability $a(\mathbf{z}) = \frac{q(\mathbf{z})}{M\pi(\mathbf{z})}$, where M is chosen so that $\pi(\mathbf{z})M \geq q(\mathbf{z}), \forall \mathbf{z}$, RS generates exact samples from $q(\mathbf{z})$. The average sampling efficiency, or acceptance rate, of this sampler is $1/M$, which means the less the proposal needs to be scaled to dominate the target, the fewer samples are rejected. In practice, it can be difficult to find the smallest valid M as $q(\mathbf{z})$ is usually a complicated multi-modal distribution.

We can match $q(\mathbf{z})$ exactly with Lars by emulating classical RS and setting $a^*(\mathbf{z}) = c\frac{q(\mathbf{z})}{\pi(\mathbf{z})}$ with the constant c chosen so that $a^*(\mathbf{z}) \leq 1$ for all $\mathbf{z}$.[2] Noting that this results in $Z = c$ and comparing this to standard RS, we find that Z plays the role of $1/M$. However, with Lars our primary goal is learning a compact and fast-to-evaluate density that approximates $q(\mathbf{z})$, rather than constructing a way to sample from $q(\mathbf{z})$ exactly, which is already easy in our setting.

[2]This assumes that a^* has unlimited capacity.

4 VAEs with resampled priors

While Lars can be used to enrich several distributions in a VAE, we will concentrate on applying it to the prior; that is, we use the original VAE prior as a proposal distribution $\pi(\mathbf{z})$ and define a *resampled prior* $p(\mathbf{z})$ through a rejection sampler with learned acceptance probability $a(\mathbf{z})$. We parameterize $a(\mathbf{z})$ with a flexible neural network and use the terms "Lars prior" and "resampled prior" interchangeably.

In general, the acceptance function is trained jointly with the other parts of the model by optimizing the ELBO objective (Eq. (1)) but with the resampled prior instead of the standard Normal prior. This only modifies the prior term $\log p(\mathbf{z})$ in the objective, with $p(\mathbf{z})$ replaced by Eq. (3) for indefinite and by Eq. (4) for truncated resampling. See Appendix B.3 for pseudo-code and further details. Thus, training a VAE with a resampled prior only requires *evaluating the resampled prior density* on samples from the variational posterior as well as generating a modest number of samples from the proposal to update the moving average estimate of Z as described above. *Crucially, we never need to perform accept/reject sampling during training.* Thus, we can choose the maximum number of resampling steps T based on computational budget *at sampling time.*

Hierarchical models. In addition to VAEs with a single stochastic layer, we also consider hierarchical VAEs with two stochastic layers. We closely follow the proposed inference and generative structure of [30] and apply the resampled prior to the top-most latent layer. Thus, the generative distribution and the variational posterior factorize as $p_\theta(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2)p_\lambda(\mathbf{z}_1|\mathbf{z}_2)p_\lambda(\mathbf{z}_2)$ and $q_\psi(\mathbf{z}_1|\mathbf{x}, \mathbf{z}_2)q_\phi(\mathbf{z}_2|\mathbf{x})$ respectively, with:

$$p_\lambda(\mathbf{z}_2) = p_T(\mathbf{z}_2) \tag{7}$$

$$p_\lambda(\mathbf{z}_1|\mathbf{z}_2) = \mathcal{N}\big(\mathbf{z}_1|\mu_\lambda(\mathbf{z}_2), \mathrm{diag}\big(\sigma^2_\lambda(\mathbf{z}_2)\big)\big) \tag{8}$$

$$q_\phi(\mathbf{z}_2|\mathbf{x}) = \mathcal{N}\big(\mathbf{z}_2|\mu_\phi(\mathbf{x}), \mathrm{diag}\big(\sigma^2_\phi(\mathbf{x})\big)\big) \tag{9}$$

$$q_\psi(\mathbf{z}_1|\mathbf{x}, \mathbf{z}_2) = \mathcal{N}\big(\mathbf{z}_1|\mu_\psi(\mathbf{x}, \mathbf{z}_2), \mathrm{diag}\big(\sigma^2_\psi(\mathbf{x}, \mathbf{z}_2)\big)\big) \tag{10}$$

The means $\mu_{\{\mu,\phi,\psi\}}$ and standard deviations $\sigma_{\{\mu,\phi,\psi\}}$ are parameterized by neural networks. We will apply resampling to the top-most prior distribution $p_\lambda(\mathbf{z}_2)$.

5 Experiments

Datasets. We perform unsupervised learning on a suite of standard datasets: static and dynamically binarized MNIST [19], Omniglot [18], and FashionMNIST [32]; for more details, see Appendix B.1.

Architectures/Models. We consider several architectures: (i) a VAE with single latent layer and an MLP encoder/decoder (referred to as VAE); (ii) a VAE

Dataset	VAE ($L = 1$) standard	VAE ($L = 1$) Lars	HVAE ($L = 2$) standard	HVAE ($L = 2$) Lars	ConvHVAE ($L = 2$) standard	ConvHVAE ($L = 2$) Lars
staticMNIST	89.11	**86.53**	85.91	**84.42**	82.63	**81.70**
dynamicMNIST	84.97	**83.03**	82.66	**81.62**	81.21	**80.30**
Omniglot	104.47	**102.63**	101.38	**100.40**	97.76	**97.08**
FashionMNIST	228.68	**227.45**	227.40	**226.68**	226.39	**225.92**

Table 1: Test negative log likelihood (NLL; *lower is better*) for different models with standard Normal prior ("standard") and our proposed resampled prior ("Lars"). L is the number of stochastic layers in the model.

Model	standard $\mathcal{N}(0,1)$ ELBO	KL	recons	Z	Lars *post-hoc* ELBO	KL	recons	Z	Lars *joint training* ELBO	KL	recons	Z
dynamicMNIST	89.0	25.8	63.2	1	87.3	24.1	63.2	0.023	86.6	24.7	61.9	0.015
Omniglot	110.8	37.2	73.6	1	108.8	35.2	73.6	0.015	108.4	35.9	72.5	0.011

Table 2: ELBO and its components (KL and reconstruction error) for VAEs with: a standard Normal prior, a post-hoc fitted Lars prior, and a jointly trained Lars prior.

with two latent layers and an MLP encoder/decoder (HVAE); (iii) a VAE with two latent layers and a convolutional encoder/decoder (ConvHVAE). For the acceptance function we use an MLP network with two layers of 100 `tanh` units and the `logistic` non-linearity for the output. Moreover, we perform resampling with truncation after $T = 100$ attempts (Eq. (4)), as a good trade-off between approximation accuracy and sampling efficiency at generation time. We explore alternative architectures and truncations below. Full details of all architectures can be found in Appendix B.2.

Training and evaluation. All models were trained using the Adam optimizer [16] for 10^7 iterations with the learning rate $3{\cdot}10^{-4}$, which was decayed to $1{\cdot}10^{-4}$ after 10^6 iterations, and mini-batches of size 128. Weights were initialized according to [7]. Unless otherwise stated, we used warm-up (KL-annealing) for the first 10^5 iterations [2]. We quantitatively evaluate all methods using the negative test log likelihood (NLL) estimated using 1000 importance samples [3]. We estimate Z using S MC samples (see Eq. (5)) with $S = 10^{10}$ for evaluation after training (this only takes several minutes on a GPU) and $S = 2^{10}$ during training, see Appendix A.4 for further details. We repeated experiments for different random seeds with very similar results, so report only the means. We observed overfitting on static MNIST, so on that dataset we performed early stopping using the NLL on the validation set and halved the times for learning rate decay and warm-up.

5.1 Quantitative Results

First, we compare the resampled prior to a standard Normal prior on several architectures, see Table 1. For all architectures and datasets considered, the resampled prior outperforms the standard prior. In most cases, the improvement in the NLL exceeds 1 nat and is notably larger for the simpler architectures.

Applied *post-hoc* to a pretrained model (fixed encoder/decoder) the resampled prior almost reaches the performance of a jointly trained model, see Table 2. This highlights that Lars can also be effectively employed to improve trained models. In this case, the gain in ELBO is entirely due to a reduction in the KL as the reconstruction error is determined by the fixed encoder/decoder. In contrast, joint training reduces both, and reaches a better ELBO, though the reduction in KL is smaller than for post-hoc training.

Model	NLL
VAE ($L = 2$) + VGP [31]	81.32
LVAE ($L = 5$) [28]	81.74
HVAE[‡] ($L = 2$) + VampPrior [30]	**81.24**
HVAE ($L = 2$) + Lars prior	81.62
VAE + IAF [15]	**79.10**
ConvHVAE[‡] + Vamp [30]	79.75
ConvHVAE + Lars prior	80.30

Table 3: Test NLL on dynamic MNIST for non-convolutional (above the line) and convolutional (below the line) models.

Model	NLL
IWAE ($L = 2$) [3]	103.38
LVAE ($L = 5$) [28]	102.11
HVAE[‡] ($L = 2$) + VampPrior [30]	101.18
HVAE ($L = 2$) + Lars prior	**100.40**
ConvHVAE[‡] ($L = 2$) + VampPrior [30]	97.56
ConvHVAE ($L = 2$) + Lars prior	97.08

Table 4: Test NLL on dynamic Omniglot.

In Tables 3 and 4 we compare Lars to other approaches on dynamic MNIST and Omniglot. Lars is competitive with other approaches that make the prior or vari-

[‡]We use the same structure but a simpler architecture than [30].

ational posterior more expressive when applied to similar architectures. We expect that our results can be further improved by using PixelCNN-style decoders, which have been used to obtained state-of-the-art results on dynamic MNIST (78.45 nats for PixelVAE with VampPrior [30]) and Omniglot (89.83 with VLAE [4]). The improvements of LARS over the standard Normal prior are generally comparable to those obtained by the VampPrior [30], though they tend to be slightly smaller. LARS priors and VampPriors have somewhat complementary strengths. Sampling from VampPriors is more efficient as it amounts to sampling from a mixture model, which is non-iterative. On the other hand, the VampPriors that achieve the above results use 500 or 1000 pseudo-inputs and thus have hundreds of thousands of parameters, whereas the number of parameters in the LARS priors is more than an order of magnitude lower. As LARS priors operate on a low-dimensional latent space they are also considerably more efficient to train than VampPriors, which are connected to the input space through the pseudo-inputs.

Influence of network architecture. We investigated different network sizes for the acceptance function. As for the illustrative example, even a simple architecture can already notably improve over a standard prior, with successively more expressive networks further improving performance, see Table 5. Notably, the average acceptance probability Z is influenced much more by the truncation parameter than the network architecture. In practice, we choose $a = \text{MLP}[100-100]$ as more expressive networks only showed a marginal improvement. While we suspect that substantially larger networks might lead to overfitting, we did not observe this for the networks considered in Table 5. The relative size of the a network w.r.t. encoder and decoder also determines the extra computational cost of LARS over a standard prior. For example, the VAE ($L = 1$) results in Table 1 require 25% more multiply-adds per iteration; due to our unoptimized implementation, the difference in wall-clock time was closer to 40%.

Influence of truncation. The truncation parameter T tunes the trade-off between sampling efficiency and approximation quality. As shown in Table 6, the NLL improves as the number of steps before truncating increases, whereas the value for Z decreases accordingly. The gains in the NLL come both from the KL as well as the reconstruction term, though the relative contribution from the KL is larger. We stress that for truncated sampling, Z denotes the average acceptance rate *for the first* $T-1$ *attempts* and does not take into account always accepting the T^{th} proposal. As a reference point, a model that is *trained* with indefinite (untruncated) resampling only accepts 2 out of 10^6 samples and estimating Z accurately thus requires a lot of samples.

MODEL	NLL	Z
VAE ($L = 1$, MLP)	84.97	1
VAE + LARS; $a = \text{MLP}[10-10]$	84.08	0.02
VAE + LARS; $a = \text{MLP}[50-50]$	83.29	0.016
VAE + LARS; $a = \text{MLP}[100-100]$	83.05	0.015
VAE + LARS; $a = \text{MLP}[50-50-50]$	83.09	0.015
VAE + LARS; $a = \text{MLP}[100-100-100]$	82.94	0.014

Table 5: Test NLL and Z on dynamic MNIST. Different network architectures for $a(\mathbf{z})$ with $T = 100$.

MODEL	NLL	Z	KL	recons
VAE ($L = 1$, MLP)	84.97	1	25.8	63.3
VAE + LARS; $p_{T=2}$	84.60	0.19	25.4	63.2
VAE + LARS; $p_{T=5}$	84.11	0.12	25.1	63.0
VAE + LARS; $p_{T=10}$	83.71	0.08	25.0	62.6
VAE + LARS; $p_{T=50}$	83.24	0.03	24.8	62.1
VAE + LARS; $p_{T=100}$	83.05	0.014	24.7	61.9
VAE + LARS; p_∞	82.67	$2 \cdot 10^{-6}$	24.8	61.5

Table 6: Influence of the truncation parameter on the test set of dynamic MNIST; $a = \text{MLP}[100-100]$.

MODEL	NLL	Z
VAE ($L = 1$) + $\mathcal{N}(0, 1)$ prior	84.97	1
VAE + RealNVP prior	81.86	1
VAE + MAF prior	81.58	1
VAE + LARS $\mathcal{N}(0, 1)$	83.04	0.015
VAE + LARS RealNVP	**81.15**	0.027

Table 7: More expressive proposal and prior distributions. Test NLL on dynamic MNIST.

Combination with non-factorial priors. As stated before, LARS is an orthogonal approach to specifying rich distributions and can be combined with other structured proposals such as non-factorial or autoregressive (flow) distributions. We investigated this using a RealNVP distribution as an alternative proposal on MNIST (Table 7) and Omniglot (Table C.1), see Appendix B.2 for details. The RealNVP prior by itself outperforms our resampling approach applied to a standard Normal prior. However, applying LARS to a RealNVP proposal distribution, we obtain a further improvement of more than 0.5 nats on both MNIST and Omniglot. A VAE trained with a more expressive Masked Autoregressive Flow (MAF) [23] as a prior outperformed the RealNVP prior but was still worse than a resampled RealNVP. Combining LARS with an MAF proposal is impractical due to slow sampling of the MAF.

5.2 Qualitative Results

Samples. For a qualitative analysis, we consider unconditional *samples* from a VAE with a resampled prior. Our learned acceptance function assigns probability values $a(\mathbf{z})$ to all points in the latent space. This allows us to effectively *rank* draws from the proposal and to com-

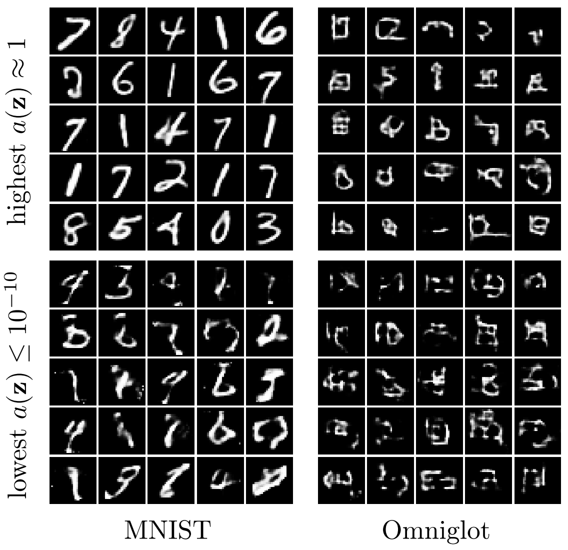
highest $a(\mathbf{z}) \approx 1$
lowest $a(\mathbf{z}) \leq 10^{-10}$
MNIST
Omniglot

Figure 3: Ranked sample means *from the proposal* of a VAE ($L = 1$, MLP) with jointly trained resampled prior. We drew 10^4 samples and show those with highest $a(\mathbf{z})$ (top) and lowest $a(\mathbf{z})$ (bottom).

Model	NLL	Z
VAE ($L = 1$) ($\mathcal{N}(0, 1)$ prior)	84.97	1
VAE (resampled $\mathcal{N}(0, 1)$ prior)	**83.04**	0.015
VAE resampled outputs ($\mathcal{N}(0, 1)$ prior)	83.63	0.014

Table 8: Test NLL on dynamic MNIST for a VAE with *resampled outputs*.

pare particularly high- to particularly low-scoring ones. We perform the same analysis twice, once for a resampled prior that is trained jointly with the VAE (Fig. 3) and once for a resampled prior that is trained post-hoc for a fixed VAE (Fig. C.7), both times with very similar results: Samples with $a(\mathbf{z}) \approx 1$ are visually pleasing and show almost no artefacts, whereas samples with $a(\mathbf{z}) \leq 10^{-10}$ look poor. When accepting/rejecting samples according to our pre-defined resampling scheme, these low-quality samples would most likely be rejected. Thus, our approach is able to (i) automatically detect (rank) samples, and (ii) reject low-quality samples. For more samples, see Appendix C.3.

Dimension-wise resampling. Instead of resampling the entire vector $\mathbf{z}$, a less wasteful sampling scheme would be to resample each dimension independently according to its own learned acceptance function $a_i(z_i)$ (or in blocks of several dimension). However, such a factorized approach did not improve over a standard (factorized) Normal prior in our case. We speculate that it might be more effective when the target density has strong factorial structure, as might be the case for β-VAEs [14]; however, we leave this as future work.

VAE with 2D latent space. In Appendix C.6 we consider a resampled VAE with 2D latent space. While the NLL is not substantially different ([12] show that the marginal KL gap is very small in this case), the resampled prior matches the aggregate posterior better and, interestingly, the jointly trained models give rise to a *broader* aggregate posterior distribution, as the resampled prior upweights the tails of the proposal.

5.3 Resampling *discrete* outputs of a VAE

So far, we have applied Lars only to the continuous variables of the relatively low-dimensional latent space of a VAE. However, our approach is more general and can also be used to resample *discrete and high-dimensional* random variables. We apply Lars directly to the 784-dimensional binarized (Bernoulli) outputs of a standard VAE in the pixel space. That is, we make the marginal likelihood richer by resampling it in the same way as the prior above: $\log p(\mathbf{x}) = \log \frac{\pi(\mathbf{x})a(\mathbf{x})}{Z}$, where $\pi(\mathbf{x})$ is now the original marginal likelihood, $a(\mathbf{x})$ is the acceptance function in the pixel space, and Z is the normalization constant. We can lower-bound the enriched marginal likelihood via its ELBO:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log \pi(\mathbf{x}|\mathbf{z}) - \mathrm{KL}(q(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})) + \log \tfrac{a(\mathbf{x})}{Z} \quad (11)$$

In practice, we use the truncated sampling density with $T = 100$ instead. $\mathbf{x}$ corresponds to the observation under consideration, whereas Z is estimated on decoded samples from the (regular) prior $p(\mathbf{z})$:

$$Z = \int \pi(\mathbf{x}|\mathbf{z})p(\mathbf{z})a(\mathbf{x})\mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z} \approx \tfrac{1}{S} \textstyle\sum_s^S a(\mathbf{x}_s), \quad (12)$$

with $\mathbf{x}_s \sim \pi(\mathbf{x}|\mathbf{z}_s)$ and $\mathbf{z}_s \sim p(\mathbf{z})$. We used the same MLP encoder/decoder as in our previous experiments, but increased the size of the $a(\mathbf{z})$ network to match the architecture of the encoder. As $\mathbf{x}$ is a discrete random variable, we cannot easily compute the contribution of $\log Z$ to the gradient for the encoder and decoder parameters due to the inapplicability of the reparameterization trick. For simplicity, we ignore this contribution, which effectively leads to post-hoc fitting of $a(\mathbf{x})$, see Appendix B.4 for a discussion.

Resampling the outputs of a VAE is slower than resampling the prior for several reasons: (i) we need to decode all prior samples $\mathbf{z}_s$, (ii) a larger $a(\mathbf{x})$ has to act on a higher dimensional space, and (iii) estimation of Z typically requires more samples to be accurate (we used $S = 10^{11}$). These are also the reasons why we apply Lars in the lower-dimensional latent space to enrich the prior in all other experiments in this paper.

Still, Lars works fairly well in this case. In terms of NLL, the VAE with resampled outputs outperforms the original VAE by over 1 nat (Table 8) and is only about 0.6 nats worse than the VAE with resampled prior. The learned acceptance function can be used to rank the binary output samples, see Fig. C.8.

6 Related Work

Expressive prior distribution for VAEs. Recently, several methods of improving VAEs by making their priors more expressive have been proposed. The VampPrior [30] is parameterized as a mixture of variational posteriors obtained by applying the encoder to a fixed number of learned pseudo-observations. While sampling from the VampPrior is fast, evaluating its density can be relatively expensive as the number of pseudo-observations used tends to be in the hundreds and the encoder needs to be applied on each one. A mixture of Gaussian prior [5, 22] provides a simpler alternative to the VampPrior, but is harder to optimize and does not perform as well [30]. Autoregressive models parameterized with neural networks allow specifying very expressive priors [e.g. 8, 10, 23] that support fast density evaluation but sampling from them is slow, as it is an inherently sequential process. While sampling in Lars also appears sequential, it can be easily parallelized by considering a large number of candidates in parallel. Chen et al. [4] used autoregressive flows to define priors which are both fairly fast to evaluate and sample from. Flow-based approaches require invertible transformations with fast-to-compute Jacobians; these requirements limit the power of any single transformation and thus necessitate chaining a number of them together. In contrast, Lars places no restriction on the parameterization of the acceptance function and can work well even with fairly small neural networks. On the other hand, evaluating models trained with Lars requires estimating the normalizing constant using a large number of MC samples. As we showed above however, Lars and flows can be combined to give rise to models with better objective value as well as higher sampling efficiency.

Rejection Sampling (RS). Variational Rejection Sampling [9] uses RS to make the variational posterior closer to the exact posterior; as it relies on evaluating the target density, it is not applicable to our setting. Naesseth et al. [21] derive reparameterization gradients for some not directly reparameterizable distributions by using RS to correct for sampling from a reparameterizable proposal instead of the distribution of interest. Similarly to VRS, this method requires being able to evaluate the target density.

Density Ratio Estimation (DRE). DRE [29] estimates a ratio of densities by training a classifier to discriminate between samples from them. It has been used to estimate the KL term in the ELBO in order to train VAEs with implicit priors [20], but the approach tends to overestimate the ELBO [26], making model comparison difficult. The acceptance probability function learned by Lars can be interpreted as estimating a (rescaled) density ratio between the aggregate posterior and the proposal, which is done end-to-end as a part of the generative process. Note that we do not aim to estimate the density ratio as accurately as possible. Instead, we aim to learn a compact model for $a(\mathbf{z})$, so that the resulting $p(\mathbf{z})$ strikes a good balance between approximating $q(\mathbf{z})$ well and generalizing to new observations. Moreover, as the resampled prior has an explicit density, we can perform reliable model comparison by estimating the normalizing constant using a large number of samples.

Products of Experts. The density induced by Lars can be seen as a special instance of the product-of-experts (PoE, [11]) architecture with two experts. However, while sampling from PoE models is generally difficult as it requires MCMC algorithms, our approach yields exact independent samples because of its close relationship to classical rejection sampling.

7 Conclusion

We introduced Learned Accept/Reject Sampling (Lars), a powerful method for approximating a target density q given a proposal density and samples from the target. Lars uses a learned acceptance function to reweight the proposal, and can be applied to both continuous and discrete variables. We employed Lars to define a resampled prior for VAEs and showed that it outperforms a standard Normal prior by a considerable margin on several architectures and datasets. It is competitive with other approaches that enrich the posterior or prior and can be efficiently combined with the ones that support efficient sampling, such as flows. Lars can also be applied post hoc to already trained models and its learned acceptance function effectively ranks samples from the proposal by visual quality.

We addressed two challenges of our approach, potentially low sampling efficiency and the requirement to estimate the normalization constant, and demonstrated that inference remains tractable and that a truncated resampling scheme provides an interpretable way to trade off sampling efficiency against approximation quality.

Lars is not limited to variational inference and exploring its potential in other contexts is an interesting direction for future research. Its successful application to resampling the discrete outputs of a VAE is a first step in this direction. We also envision generalizing our approach to more than one acceptance function, e.g. resampling dimensions independently, or using a different resampling scheme for the first T and for the following attempts.

Acknowledgements

We thank Ulrich Paquet, Michael Figurnov, Guillaume Desjardins, Mélanie Rey, Jörg Bornschein, Hyunjik Kim, and George Tucker for their helpful comments and suggestions.

References

[1] Aleksandar Botev, Bowen Zheng, and David Barber. "Complementary Sum Sampling for Likelihood Approximation in Large Scale Classification". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. 2017.

[2] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. "Generating Sentences from a Continuous Space". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 2016.

[3] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance weighted autoencoders". In: *Proceedings of the 4th International Conference on Learning Representations*. 2016.

[4] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Variational Lossy Autoencoder". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017.

[5] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. "Deep unsupervised clustering with gaussian mixture variational autoencoders". In: *arXiv preprint arXiv:1611.02648* (2016).

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017.

[7] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010.

[8] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. "DRAW: A Recurrent Neural Network For Image Generation". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[9] Aditya Grover, Ramki Gummadi, Miguel Lazaro-Gredilla, Dale Schuurmans, and Stefano Ermon. "Variational Rejection Sampling". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. 2018.

[10] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. "PixelVAE: A Latent Variable Model for Natural Images". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017.

[11] Geoffrey Hinton. "Training Products of Experts by Minimizing Contrastive Divergence". In: *Neural Computation* (2000).

[12] Matthew D. Hoffman and Matthew J. Johnson. "ELBO Surgery". In: *NIPS 2016 Workshop on Advances in Approximate Bayesian Inference* (2016).

[13] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. "Neural Autoregressive Flows". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[14] Arka Pal Christopher Burgess Xavier Glorot Matthew Botvinick Shakir Mohamed Alexander Lerchner Irina Higgins Loic Matthey. "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *Proceedings of the 5th International Conference on Learning Representations*. 2017.

[15] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improved Variational Inference with Inverse Autoregressive Flow". In: *Advances in Neural Information Processing Systems 29*. 2016.

[16] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the 3rd International Conference on Learning Representations*. 2015.

[17] Diederik Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *Proceedings of the 2nd International Conference on Learning Representations*. 2014.

[18] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 6266 (2015).

[19] Hugo Larochelle and Iain Murray. "The Neural Autoregressive Distribution Estimator". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011.

[20] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. "Adversarial Variational Bayes: Unifying Variational Autoencoders and Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017.

[21] Christian Naesseth, Francisco Ruiz, Scott Linderman, and David Blei. “Reparameterization Gradients through Acceptance-Rejection Sampling Algorithms”. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. 2017.

[22] Eric Nalisnick, Lars Hertel, and Padhraic Smyth. “Approximate inference for deep latent gaussian mixtures”. In: *NIPS Workshop on Bayesian Deep Learning*. 2016.

[23] George Papamakarios, Iain Murray, and Theo Pavlakou. “Masked Autoregressive Flow for Density Estimation”. In: *Advances in Neural Information Processing Systems 30*. 2017.

[24] Danilo Jimenez Rezende and Shakir Mohamed. “Variational Inference with Normalizing Flows”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning*. 21, 2015.

[25] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014.

[26] Mihaela Rosca, Balaji Lakshminarayanan, and Shakir Mohamed. “Distribution Matching in Variational Inference”. In: *arXiv* (2018).

[27] Tim Salimans, Diederik Kingma, and Max Welling. “Markov Chain Monte Carlo and Variational Inference: Bridging the gap”. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[28] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. “Ladder Variational Autoencoders”. In: *Advances in Neural Information Processing Systems 29*. 2016.

[29] M. Sugiyama, T. Suzuki, and T. Kanamori. “Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation”. In: *Annals of the Institute of Statistical Mathematics* 5 (2012).

[30] Jakub Tomczak and Max Welling. “VAE with a VampPrior”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. 2018.

[31] Dustin Tran, Rajesh Ranganath, and David M. Blei. “The Variational Gaussian Process”. In: *Proceedings of the 4th International Conference on Learning Representations*. 2016.

[32] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 28, 2017.

[33] John von Neumann. “Various techniques used in connection with random digits”. In: *Monte Carlo Method*. 1951.