Generative Modeling with Flow-Guided Density Ratio Learning

Alvin Heng[1], Abdul Fatir Ansari[2,4], and Harold Soh[1,3] (✉)

[1] Department of Computer Science, National University of Singapore, Singapore
`{alvinh,harold}@comp.nus.edu.sg`
[2] AWS AI Labs, Berlin, Germany
[3] Smart Systems Institute, National University of Singapore, Singapore
[4] Work done while at the National University of Singapore, prior to joining Amazon

Abstract. We present Flow-Guided Density Ratio Learning (FDRL), a simple and scalable approach to generative modeling which builds on the stale (time-independent) approximation of the gradient flow of entropy-regularized f-divergences introduced in recent work. Specifically, the intractable time-dependent density ratio is approximated by a stale estimator given by a GAN discriminator. This is sufficient in the case of sample refinement, where the source and target distributions of the flow are close to each other. However, this assumption is invalid for generation and a naive application of the stale estimator fails due to the large chasm between the two distributions. FDRL proposes to train a density ratio estimator such that it learns from progressively improving samples during the training process. We show that this simple method alleviates the density chasm problem, allowing FDRL to generate images of dimensions as high as 128×128, as well as outperform existing gradient flow baselines on quantitative benchmarks. We also show the flexibility of FDRL with two use cases. First, unconditional FDRL can be easily composed with external classifiers to perform class-conditional generation. Second, FDRL can be directly applied to unpaired image-to-image translation with no modifications needed to the framework. Our code and relevant supplementary material are available at `https://github.com/clear-nus/fdrl`.

Keywords: generative models · image generation · gradient flows.

1 Introduction

Deep generative modeling has made significant strides in recent years. Advances in techniques such as generative adversarial networks [13] and diffusion models [15,33] have enabled the generation of high-quality, photorealistic images and videos. Recently, gradient flows have emerged as an interesting alternative approach to generative modeling. Unlike the aforementioned techniques, gradient flows have a unique interpretation where the objective is to find the path of steepest descent between two distributions. Further, where diffusion models typically require a tractable terminal distribution such as Gaussian, the source and

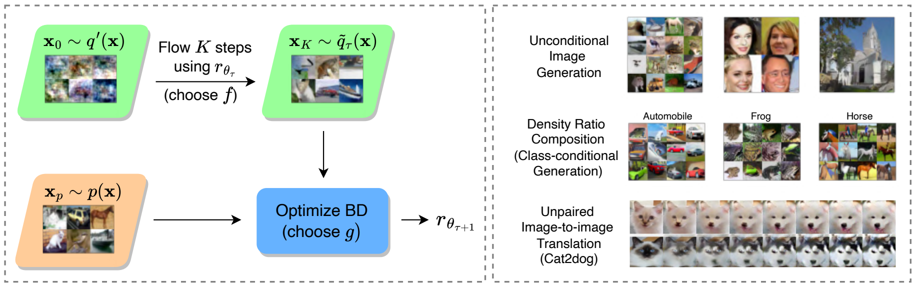

$\mathbf{x}_0 \sim q'(\mathbf{x})$
Flow K steps using r_{θ_τ}
(choose f)
$\mathbf{x}_K \sim \tilde{q}_\tau(\mathbf{x})$
$\mathbf{x}_p \sim p(\mathbf{x})$
Optimize BD (choose g)
$r_{\theta_{\tau+1}}$
Unconditional Image Generation
Density Ratio Composition (Class-conditional Generation)
Automobile
Frog
Horse
Unpaired Image-to-image Translation (Cat2dog)

Fig. 1: Left: Illustration of FDRL's training setup for the τ training iteration. For clarity, we emphasize the choices of the f-divergence and g, the Bregman divergence function, as part of training. Right: the various applications of FDRL, ranging from unconditional image generation to class-conditional generation by composition with external classifiers and unpaired image-to-image translation.

target distributions in gradient flows can be varied for different tasks in generative modeling. For instance, the gradient flow performs data synthesis when the source distribution is a simple prior and the target is the data distribution. When the two distributions are from different domains of the same modality, the gradient flow performs domain translation.

Among gradient flow methods, Wasserstein gradient flows (WGF) have recently become a popular specialization for a variety of applications [22,9,2,11]. WGFs model the gradient dynamics on the space of probability measures with respect to the Wasserstein metric; these methods aim to construct the optimal path between two probability measures — a source distribution $q(\mathbf{x})$ and a target distribution $p(\mathbf{x})$, where the notion of optimality refers to the path of steepest descent in Wasserstein space.

Despite the immense potential of gradient flow methods in generative modeling [2,11,9], applications to high-dimensional image synthesis remain limited, and its applications to other domains, such as class-conditional generation and image-to-image translation have been underexplored. A key difficulty is that existing methods to solve for the gradient flow often involve complex approximation schemes. For instance, the JKO scheme [16] necessitates estimation of the 2-Wasserstein distance and divergence functionals, while particle simulation methods, which involve simulating an equivalent stochastic differential equation (SDE), requires solving for the time-dependent marginal distribution over the flow.

A recent work on adopting gradient flows for sample refinement, DG$f$low [2], proposes a stale (time-independent) estimate given by the discriminator of a pretrained GAN, as the density ratio estimate necessary in the corresponding SDE. While this dramatically simplifies the particle simulation approach, in practice this only works when $q(\mathbf{x})$ and $p(\mathbf{x})$ are close together, such as between the GAN generated images and the true data distribution. When the two distributions are far apart, such as the case of image synthesis, where $q(\mathbf{x})$ is a simple prior distribution, estimating the density ratio $q(\mathbf{x})/p(\mathbf{x})$ becomes a trivial problem due to the large density chasm [27], and use of this naive stale estimator in simulating the SDE fails to generate realistic images.

We therefore pose the question: is it possible to adopt a simple gradient flow scheme for image generation driven by density ratio learning? We answer this in the affirmative by proposing Flow-Guided Density Ratio Learning (FDRL), a new training approach to address the problem of estimating the density ratio between the prior and the data distribution, by progressively training a density ratio estimator with evolving samples. FDRL operates exclusively in the data space, and does not require training additional generators, unlike related particle methods [11]. Compared to gradient flow baselines, we achieve the best quantitative scores in image synthesis and are the first to scale to image dimensions as high as 128×128. In addition, we show that our framework can be seamlessly applied to two other critical tasks in generative modeling: class-conditional generation and unpaired image-to-image translation, which have been under-explored in the context of gradient flow methods.

2 Background

We provide a brief overview of gradient flows and density ratio estimation. For a more comprehensive introduction to gradient flows, please refer to [29]. A thorough overview of density ratio estimation can be found in [34].

2.1 Wasserstein Gradient Flows.

Consider Euclidean space equipped with the familiar L_2 distance metric $(\mathcal{X}, \|\cdot\|_2)$. Given a function $F : \mathcal{X} \to \mathbb{R}$, the curve $\{\mathbf{x}(t)\}_{t\in\mathbb{R}^+}$ that follows the direction of steepest descent is called the *gradient flow* of F:

$$\mathbf{x}'(t) = -\nabla F(\mathbf{x}(t)). \tag{1}$$

In generative modeling, we are interested in sampling from the underlying *probability distribution* of a given dataset. Hence, instead of Euclidean space, we consider the space of *probability measures* with finite second moments equipped with the 2-Wasserstein metric $(\mathcal{P}_2(\Omega), \mathcal{W}_2)$. Given a functional $\mathcal{F} : \mathcal{P}_2(\Omega) \to \mathbb{R}$ in the 2-Wasserstein space, the gradient flow of $\mathcal{F}$ is the steepest descent curve of $\mathcal{F}$. Such curves are called Wasserstein gradient flows.

2.2 Density Ratio Estimation via Bregman Divergence.

Let $q(\mathbf{x})$ and $p(\mathbf{x})$ be two distributions over $\mathcal{X} \in \mathbb{R}^d$ where we have access to i.i.d samples $\mathbf{x}_q \sim q(\mathbf{x})$ and $\mathbf{x}_p \sim p(\mathbf{x})$. The goal of density ratio estimation (DRE) is to estimate the true density ratio $r^*(\mathbf{x}) = \frac{q(\mathbf{x})}{p(\mathbf{x})}$ based on samples $\mathbf{x}_q$ and $\mathbf{x}_p$.

We will focus on density ratio fitting under the Bregman divergence (BD), which is a framework that unifies many existing DRE techniques [34,35]. Let g :

$\mathbb{R}^+ \to \mathbb{R}$ be a twice continuously differentiable convex function with a bounded derivative. The BD seeks to quantify the discrepancy between the estimated density ratio r_θ and the true density ratio r^*:

$$BD_g(r^*||r_\theta) = \mathbb{E}_p[g(r^*(\mathbf{x})) - g(r_\theta(\mathbf{x})) + \partial g(r_\theta(\mathbf{x}))r_\theta(\mathbf{x})] - \mathbb{E}_q[\partial g(r_\theta(\mathbf{x}))]. \quad (2)$$

In practice, we estimate the expectations in Eq. 2 using Monte Carlo samples. We can also drop the term $\mathbb{E}_p[g(r^*(\mathbf{x}))]$ during optimization as it does not depend on the model r_θ.

The minimizer of Eq. 2, which we denote θ^*, satifies $r_{\theta^*}(x) = r^*(\mathbf{x}) = q(\mathbf{x})/p(\mathbf{x})$. Moving forward, we will use the hatted symbol $\widehat{\mathbb{E}}_p$ to refer to Monte Carlo estimates for ease of notation. We consider two common instances of g in this paper, namely $g(y) = \frac{1}{2}(y-1)^2$ and $g(y) = y \log y - (1+y)\log(1+y)$, which correspond to the Least-Squares Importance Fitting (LSIF) and Logistic Regression (LR) objectives. We provide the full form of the BD for these choices of g in the supplementary.

3 Discriminator Gradient Flow

Ansari et al. propose DG$f$low [2] for sample refinement by simulating the SDE:

$$d\mathbf{x}_t = -\nabla_{\mathbf{x}} f'(q_t(\mathbf{x}_t)/p(\mathbf{x}_t))dt + \sqrt{2\gamma}d\mathbf{w}_t, \quad (3)$$

where $d\mathbf{w}_t$ denotes the standard Wiener process. $p(\mathbf{x})$ represents the target distribution of refinement, while $q_t(\mathbf{x})$ represents the time-evolved source distribution $q_0(\mathbf{x})$. In theory, as $t \to \infty$, the stationary distribution of the particles $\mathbf{x}_t$ from Eq. 3 converges to the target distribution $p(\mathbf{x})$. The SDE is the equivalent particle system of the corresponding Fokker-Planck equation, which is the gradient flow of the functional

$$\mathcal{F}_p^f(q) = \int p(\mathbf{x}) f(q(\mathbf{x})/p(\mathbf{x}))d\mathbf{x} + \gamma \int q(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{x}, \quad (4)$$

where $f : \mathbb{R}^+ \to \mathbb{R}$ is a twice-differentiable convex function with $f(1) = 0$. Eq. 3 can therefore be understood as the flow which minimizes Eq. 4. The first term in Eq. 4 is the f-divergence, which measures the discrepancy between $q(\mathbf{x})$ and $p(\mathbf{x})$. Popular f-divergences include the Kullback-Leibler (KL), Pearson-χ^2 divergence and Jensen-Shannon (JS) divergence. Meanwhile, the second term is a differential entropy term that improves expressiveness of the flow. We list the explicit forms of f that will be considered in this work in Table 1.

In DG$f$low, the source distribution is the distribution of samples from the GAN generator, while the target distribution is the true data distribution. The time-varying density ratio $q_t(\mathbf{x})/p(\mathbf{x})$ is approximated by a stale estimate $r_\theta(\mathbf{x})$

$$d\mathbf{x}_t = -\nabla_{\mathbf{x}} f'(r_\theta(\mathbf{x}_t))dt + \sqrt{2\gamma}d\mathbf{w}_t, \quad (5)$$

where r_θ is represented by the GAN discriminator. The assumption underlying this approach is that the distribution of generated samples from the GAN is

Table 1: f-divergences and their first derivatives f'.

f-divergence	f	f'
Pearson-χ^2	$(r-1)^2$	$2(r-1)$
KL	$r\log r$	$\log r + 1$
JS	$r\log r - (r+1)\log\frac{r+1}{2}$	$\log\frac{2r}{r+1}$
log D	$(r+1)\log(r+1) - 2\log 2$	$\log(r+1)+1$

sufficiently close to the data distribution, so that the density ratio $q_t(\mathbf{x})/p(\mathbf{x})$ is approximately constant throughout the flow.

As we are unable to compute the marginal distributions $q_t(\mathbf{x})$ or estimate the time-dependent density ratio $q_t(\mathbf{x})/p(\mathbf{x})$ in Eq. 3 a priori, we are motivated to extend the stale formulation of DG$f$low from refinement to the more difficult problem of synthesis, i.e., generating samples from noise. We show that in such a scenario, the stationary distribution of Eq. 5 shares the same MLE as the data distribution.

Lemma 1. *Let $\gamma = 1$ and assume that $q(\mathbf{x}) \sim U[a,b]$, where a, b are chosen appropriately (e.g., [-1,1] for image pixels). Then the stationary distribution of Eq. 5, $\rho_\infty(\mathbf{x})$, has the same maximum likelihood estimate as $p(\mathbf{x})$, $\arg\max_{\mathbf{x}} \rho_\infty(\mathbf{x}) = \arg\max_{\mathbf{x}} p(\mathbf{x})$.*

Based on Lemma 1, we hypothesize that samples drawn from Eq. 5 have high likelihoods under the data distribution. This motivates us to train a stale estimator $r_\theta(\mathbf{x})$ where $q(\mathbf{x})$ is a simple prior and $p(\mathbf{x})$ is the true data distribution.

3.1 Where The Stale Estimator Breaks Down

This can be done with samples from $q(\mathbf{x})$ and $p(\mathbf{x})$ by leveraging the Bregman divergence of Eq. 2. However, this fails empirically as the density ratio estimation problem becomes trivial and the Bregman loss diverges. We convey the key intuitions with a toy example of simple 2D Gaussians. Consider an SDE of the form

$$d\mathbf{x}_t = -\nabla_{\mathbf{x}} f'(q(\mathbf{x}_t)/p(\mathbf{x}_t))dt + \sqrt{2\gamma}d\mathbf{w}_t, \tag{6}$$

where $q(\mathbf{x})$ and $p(\mathbf{x})$ are both simple 2D Gaussian densities. This differs from Eq. 3 as the distribution $q(\mathbf{x}_t)$ is time-independent, and fixed as the source Gaussian distribution. We would like to investigate how such an SDE, with q/p represented by a stale estimator r_θ, behaves in transporting particles from $q(\mathbf{x})$ to $p(\mathbf{x})$.

Although we have access to the analytical form of the density ratio $q(\mathbf{x})/p(\mathbf{x})$ in this example, we choose to train a multilayer perceptron (MLP) to estimate the density ratio using the Bregman divergence Eq. 2. This is to evaluate the general case where we may only have samples and not the analytic densities. We investigate two scenarios by varying the distance between q and p, as measured by their KL divergence (or more generally, the f-divergences). $q(\mathbf{x})$ is fixed at

$\mathcal{N}(\mathbf{0}, 0.1\mathbb{I})$, where bold numbers represent 2D vectors. We set $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{1}, 0.1\mathbb{I})$ in the first case and $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{6}, 0.1\mathbb{I})$ in the second.

We first train separate MLPs to estimate the density ratio, using $\mathbf{x}_q$ and $\mathbf{x}_p$ drawn from the respective distributions. After convergence, we simulate a finite number of steps of Eq. 5 on samples drawn from $q(\mathbf{x})$, with the density ratio given by the MLP. We characterize the transition of particles through Eq. 5 as parameterized by a generalized kernel

$$\tilde{q}(\mathbf{x}) = \int q(\mathbf{x}')M_\theta(\mathbf{x}|\mathbf{x}')d\mathbf{x}', \tag{7}$$

where M_θ is the transition kernel parameterized by the MLP. The results are shown in Fig. 2. In (a), where the two distributions are close together, the particles from $q(\mathbf{x})$ are successfully transported to $p(\mathbf{x})$. Meanwhile in (b), even after flowing for many more steps, the particles have not converged to the target distribution. Flowing for more steps in (b) results in the particles drifting even further from the target (and out of the image frame). This suggests that the stale estimator is only appropriate in the case where the two distributions are not too far apart, which indeed corresponds to the assumption of DG*f*low that the GAN generated images are already close to the data distribution.

This observation can be attributed to two complementary explanations. Firstly, the Bregman divergence is optimized using Monte Carlo samples; when the two distributions are separated by large regions of vanishing likelihood, there are insufficient samples in these regions for the model to learn an accurate estimate of the density ratio. Consequently, the inaccurate estimate causes the particles to drift away from the correct direction of the target as it crosses these regions, as seen in Fig. 2b.

The second explanation may be attributed to the density chasm problem [27]. When two distributions are far apart, a binary classifier (which is equivalent to a density ratio estimator, see supplementary) can obtain near perfect accuracy while learning a relatively poor estimate of the density ratio. For example, in the second case above, the neural network can trivially separate the two modes with a multitude of straight lines.

The eventual goal of this work is the generative modeling of images, where we set our source distribution to be a simple prior distribution, and the target distribution to be the distribution of natural images. In such cases involving high-dimensional distributions, the density chasm problem is dramatically more pronounced, causing the stale estimator to fail to generate realistic images. As mentioned previously, in early experiments where we attempt to learn a density ratio estimator with $q(\mathbf{x})$ being a uniform prior and $p(\mathbf{x})$ represented by samples of natural images (e.g. CIFAR10), the Bregman loss diverges early during training, leading to failed image samples.

3.2 Data-Dependent Priors

One approach to address the density chasm problem for images is to devise a prior $q(\mathbf{x})$ that is as close to the target distribution as possible, while remain-

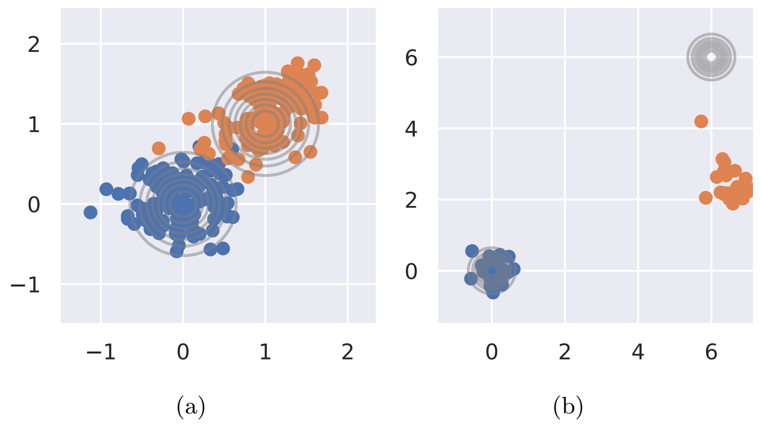


Fig. 2: Toy experiments by simulating the flow Eq. 5 with an MLP density ratio estimator. The source distribution $q(\mathbf{x})$ is set as $\mathcal{N}(\mathbf{0}, 0.1\mathbb{I})$. Blue particles represent the source particles, and orange particles represent the same particles after flowing for K steps. (a) We set $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{1}, 0.1\mathbb{I})$ and flow for $K = 15$ steps. (b) We set $p(\mathbf{x}) \sim \mathcal{N}(\mathbf{6}, 0.1\mathbb{I})$ and flow for $K = 400$ steps. We can clearly see that particles in (a) have converged to the target distribution, while particles in (b) have not, demonstrating the density chasm problem.

ing easy to sample from. Inspired by generation from seed distributions with robust classifiers [30], we first adopted a data-dependent prior (DDP) by fitting a multivariate Gaussian to the training dataset:

$$\begin{aligned} &q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_D, \boldsymbol{\Sigma}_D), \text{where} \\ &\qquad \boldsymbol{\mu}_D = \mathbb{E}_D[\mathbf{x}], \quad \boldsymbol{\Sigma}_D = \mathbb{E}_D[(\mathbf{x} - \boldsymbol{\mu}_D]^T(\mathbf{x} - \boldsymbol{\mu}_D)] \end{aligned} \tag{8}$$

where subscript D represents the dataset. Samples from this distribution contain low frequency features of the dataset, such as common colors and outlines, which we visualize in the supplementary. We proceeded to train a CNN with the Bregman divergence in the same manner as the toy experiments, with $\mathbf{x}_q$ drawn from the DDP and $\mathbf{x}_p$ from the dataset. However, we found that the Bregman loss still diverged early during training — the DDP alone is insufficient to bridge the density chasm. In the next section, we describe a novel approach to cross the chasm using *flow-guided* training, which when used with DDP achieves significantly better generative performance.

4 Flow-Guided Density Ratio Learning

We propose FDRL, a training scheme that is able to scale to the domain of high-dimensional images, while retaining the simplicity of the stale formulation of DG$f$low. Instead of merely training r_θ to estimate $q(\mathbf{x})/p(\mathbf{x})$, we propose to estimate $\tilde{q}(\mathbf{x})/p(\mathbf{x})$ by flowing samples from $q(\mathbf{x})$ at each training step with Eq. 5. This approach is based on observations that $\tilde{q}(\mathbf{x})$ progressively approaches the target distribution as r_θ is being trained. In this formulation, $\tilde{q}(\mathbf{x})$ is no longer static but changes with r_θ as θ is updated. Therefore, unlike the toy examples of Sec. 3.1, the density ratio estimator is being trained on samples that improve with training.

We formulate our training setup as follows. Consider a given training step τ, where our model has parameters θ_τ from the gradient update of the previous iteration. We switch notations slightly and denote $q'(\mathbf{x})$ as our source distribution, which is a simple prior (such as the DDP). We propose to first draw samples $\mathbf{x}_0 \sim q'(\mathbf{x})$, and simulate Eq. 5 for K steps with the stale estimator r_{θ_τ} using the Euler-Maruyama discretization:

$$\mathbf{x}_{k+1} = -\eta \nabla_{\mathbf{x}} f'(r_{\theta_\tau})(\mathbf{x}_k) + \nu \boldsymbol{\xi}_k \tag{9}$$

where $\boldsymbol{\xi}_k \sim \mathcal{N}(0, I)$, η is the step size and $k \in [0, K-1]$. We label the resultant particles as $\mathbf{x}_K$. The $\mathbf{x}_K$ are drawn from the distribution $\tilde{q}_\tau(\mathbf{x}_K)$ given by

$$\tilde{q}_\tau(\mathbf{x}_K) = \int q'(\mathbf{x}') M_{\theta_\tau}(\mathbf{x}_K|\mathbf{x}') d\mathbf{x} \tag{10}$$

where $M_{\theta_\tau}(\mathbf{x}_K|\mathbf{x})$ is the transition kernel of simulating Eq. 9 with parameters θ_τ. We optimize $r_{\theta_\tau}(\mathbf{x})$ for the τ training iteration using the Bregman divergence

$$\mathcal{L}(\theta_\tau) = \widehat{\mathbb{E}}_p[\partial g(r_{\theta_\tau}(\mathbf{x}))r(\mathbf{x}) - g(r_{\theta_\tau}(\mathbf{x}))] - \widehat{\mathbb{E}}_{\tilde{q}_\tau}[\partial g(r_{\theta_\tau}(\mathbf{x}))] \tag{11}$$

where expectation over $p(\mathbf{x})$ can be estimated using samples from the dataset. We update the parameters for the next training step $\tau+1$ as $\theta_{\tau+1} \leftarrow \theta_\tau - \alpha \nabla_\theta \mathcal{L}(\theta_\tau)$ with learning rate α. The process repeats until convergence. We provide pseudocode in Algorithm 1, and an illustration of the training setup in Fig. 1. We illustrate our training process in Fig. 3, where we utilize the toy example from Fig. 2b which the stale estimator $r_\theta = q'/p$ failed previously. We can see from the top row that as τ increases, the orange samples from $\tilde{q}_\tau$, which are obtained by flowing the blue samples from q' for fixed K steps, progressively approach the target distribution. In the bottom row, we plot the progression of the mean of $\tilde{q}_\tau$ as training progresses, which shows the iterative approach towards the target distribution. Together with the experiments for high-dimensional image synthesis in Sec. 6, these results empirically support the notion that FDRL overcomes the density chasm problem, allowing the stale estimator $r_\theta(\mathbf{x})$ to be used for sample generation.

4.1 Convergence Distribution of FDRL

In this section, we analyze the convergence distribution after training has converged. With T training steps and $\tau \in [0, T-1]$, let us consider the final training iteration $T-1$. The model has parameters θ_{T-1} from the gradient update of the previous iteration. Following Algorithm 1, we first sample $\mathbf{x}_K$ from the distribution $\tilde{q}_{T-1}(\mathbf{x}_K) = \int q'(\mathbf{x}') M_{\theta_{T-1}}(\mathbf{x}_K|\mathbf{x}') d\mathbf{x}'$ by running K steps of Eq. 9.

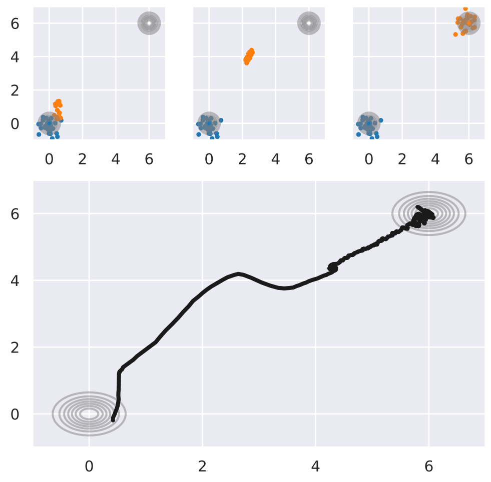


Fig. 3: Plot of $\tilde{q}_\tau$ as training progresses in the toy example of Fig. 2b. The total number of training steps is $T = 1000$, such that $\tau \in [0, T-1]$. In the top row, blue particles are samples from q' while orange particles are samples from $\tilde{q}_\tau$ at specific training steps. Left: $\tau = 10$, center: $\tau = 40$, right: $\tau = 1000$. In the bottom row, we plot the trajectory of the mean of $\tilde{q}_\tau$ as training progresses.

With samples $\mathbf{x}_K \sim \tilde{q}_{T-1}(\mathbf{x})$ and $\mathbf{x}_p \sim p(\mathbf{x})$, we optimize the BD and take the gradient step $\theta_T \leftarrow \theta_{T-1} - \alpha\nabla_\theta\mathcal{L}$. Assuming convergence, this means by definition that the loss gradient of the final step $\nabla_\theta\mathcal{L} = 0$. Thus, with no further gradient updates possible, our model is parameterized by θ_T after convergence. We can now analyze what happens when sampling from our trained model with parameters θ_T.

In our framework, we sample from our trained model by flowing for $K + \kappa$ steps. This can be viewed as two stages (K "bridging" steps, followed by κ refinement steps). To elaborate, in the first stage, we sample as before by running Eq. 9 for K steps using our trained model r_{θ_T}. The samples from stage 1 will be drawn from $\tilde{q}_T(\mathbf{x}_K) = \int q'(\mathbf{x}')M_{\theta_T}(\mathbf{x}_K|\mathbf{x}')d\mathbf{x}'$. Based on empirical evidence from our toy example in Fig. 3 and image experiments in Sec. 6.2, $\tilde{q}_T(\mathbf{x}_K)$ is close to $p(\mathbf{x})$. Interestingly, we can introduce a second refinement stage. Recall that convergence in the Bregman divergence, $\nabla_\theta\mathcal{L} = 0$, implies that our density ratio estimator has converged to the value $r_{\theta_T}(\mathbf{x}) = \tilde{q}_{T-1}/p$. Since after the first stage, we have samples from $\tilde{q}_T(\mathbf{x})$, and an estimator $r_{\theta_T}(\mathbf{x}) = \tilde{q}_{T-1}/p \approx \tilde{q}_T/p$ (assuming the change in θ in the final training step is small), we can perform refinement as per the formalism of DG$f$low by running the same Eq. 9 for a further κ steps. We provide pseudocode for sampling in Algorithm 2 and justify the two-stage sampling scheme with ablations in Sec. 6.3.

Algorithm 1 Training

```
repeat
    Sample $\mathbf{x}_p \sim p(\mathbf{x}), \mathbf{x}_0 \sim q'(\mathbf{x})$
    for $j \leftarrow 1, K$ do
        Obtain $\mathbf{x}_K$ from $\mathbf{x}_0$ by simulating Eq. 9.
    end for
    Update $\theta$ according to
        $$\nabla_\theta [g'(r_\theta(\mathbf{x}_p)) r(\mathbf{x}_p) - g(r_\theta(\mathbf{x}_p)) - g'(r_\theta(\mathbf{x}_K))]$$
until converged
```

Algorithm 2 Sampling

```
Sample $\mathbf{x}_0 \sim q'(\mathbf{x})$
for $j \leftarrow 1, K + \kappa$ do
    Obtain $\mathbf{x}_{K+\kappa}$ from $\mathbf{x}_0$ by simulating Eq. 9.
end for
return $\mathbf{x}_{K+\kappa}$
```

5 Related Works

Gradient flows are a general framework for constructing the steepest descent curve of a given functional, and consequently have been used in optimizing a variety of distance metrics, ranging from the f-divergence [12,11,2,9], maximum mean discrepancy [3,23], Sobolev distance [24] and related forms of the Wasserstein distance [20].

Two well-known methodologies to simulate Wasserstein gradient flows are population density approaches such as the JKO scheme [16] and particle-based approaches. The former requires computation of the 2-Wasserstein distance and a free energy functional, which are generally intractable. As such, several works have proposed approximations leveraging Brenier's theorem [22], Fenchel conjugate [9] and neural forward and backward schemes [1].

On the other hand, particle approaches involve simulating the SDE Eq. 3. DG$f$low leverages GAN discriminators as a stale density ratio estimator for sample refinement. Our method is inspired by the stale formulation of DG$f$low, but we extend it to the general case of sample generation. In fact, DG$f$low is a specific instance of FDRL where we fix the prior q' to be the implicit distribution defined by the GAN generator. Euler Particle Transport (EPT) [11] similarly trains a deep density ratio estimator using the Bregman divergence. However, EPT was only shown to be effective in the latent space and thus necessitated the training of an additional generator network, whereas our method is able to scale directly in the data space. Interestingly, recent work [10] has shown that particle approaches like FDRL, which train networks to estimate certain functionals (e.g., density ratio or score [31]), can be unified with methods that explicitly train generator networks such as GANs [13]. This view supports the methodology of FDRL, where the density ratio estimator is trained as a density discriminator but used as a generator during inference.

Another related line of work are energy-based models (EBMs), which adopt Langevin dynamics to sample from an unnormalized distribution [17,38]. The training scheme of FDRL bears similarity with that of Short-run EBM [25], where a sample is drawn by restarting the Langevin chain and running K gradient steps. However, FDRL is fundamentally distinct from EBMs; the former is trained with density ratio estimation, while the latter is trained with maximum likelihood. Further, it can be shown that Langevin dynamics is in fact a special case of the KL gradient flow [16,19].

6 Experiments

The goal of our experiments is to compare FDRL against gradient flow baselines. However, for completeness, we also include results from other modeling approaches, including EBMs, GANs, and Diffusion models.

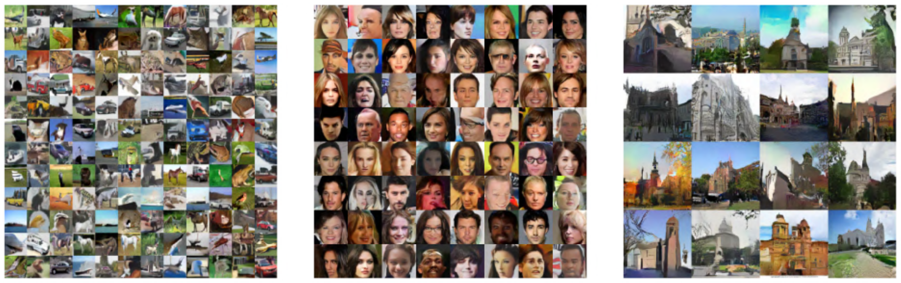


Fig. 4: Samples from FDRL-DDP on CIFAR10 32^2, CelebA 64^2 and LSUN Church 128^2 using LSIF-χ^2. More results using various BD objectives and f-divergences can be found in the supplementary.

6.1 Setup

We test FDRL with unconditional generation on 32×32 CIFAR10, 64×64 CelebA and 128×128 LSUN Church datasets, class-conditional generation on CIFAR10 and image-to-image translation on 64×64 Cat2dog dataset. We label experiments with the DDP as FDRL-DDP, and ablation experiments with a uniform prior as FDRL-UP. We use modified ResNet architectures for all experiments in this study. See the supplementary for more details.

6.2 Unconditional Image Generation

In Fig. 4, we show uncurated samples of FDRL-DDP on different combinations of g and f-divergences, where the combinations are chosen due to numerical

Method	**CIFAR10**	**CelebA**
EBM-based		
IGEBM [7]	40.58	-
Short-run EBM [25]	44.50	23.02
LEBM [26]	70.15	37.87
CoopNets [37]	33.61	-
VAEBM [36]	12.19	5.31
Diffusion-based		
NCSN [31]	25.32	26.89
DDPM [15]	3.17	-
GAN-based		
WGAN-GP [14]	55.80	30.00
SNGAN [21]	21.70	-
Gradient flow-based		
EPT [12]	24.60	-
JKO-Flow [9]	23.70	-
ℓ-SWF [6]	59.70	38.30
Ours		
FDRL-DDP (LSIF-χ^2	22.28	17.77
FDRL-DDP (LR-KL)	22.61	18.09
FDRL-DDP (LR-JS)	22.80	-
FDRL-DDP (LR-logD)	22.82	-
FDRL-UP (LSIF-χ^2	30.23	-
FDRL-UP (LR-KL)	31.99	-

Table 2: FID scores for 32×32 CIFAR10 and 64×64 CelebA. Lower is better.

compatibility (see supplementary). Visually, our model is able to produce high-quality samples on a variety of datasets up to resolutions of 128×128, surpassing existing gradient flow techniques [11,9]. Samples with other f-divergences can be found in the supplementary. In Table 2, we show the FID scores of FDRL-DDP in comparison with relevant baselines utilizing different generative approaches.

We emphasize that our experiments are geared towards comparisons with gradient flow baselines, in which FDRL-DDP outperforms both EPT and JKO-Flow, as well as the non-parametric method ℓ-SWF. Our method outperforms all baseline EBMs except VAEBM, which combines a VAE and EBM for improved performance. We also outperform WGAN and the score-based NCSN, while matching SNGAN's performance. Comparing method approaches, modern diffusion models (DDPM) outperform EBM and gradient flow methods, including FDRL. Interesting future work could investigate reasons for this gap, which may lead to improvements for EBMs and gradient flows.

To intuit the flow process, we show how samples evolve with the LSUN Church dataset in Fig. 5, where the prior contains low frequency features such as the rough silhouette of the building, and the flow process generates the higher frequency details to create a realistic image. We also visualize samples obtained

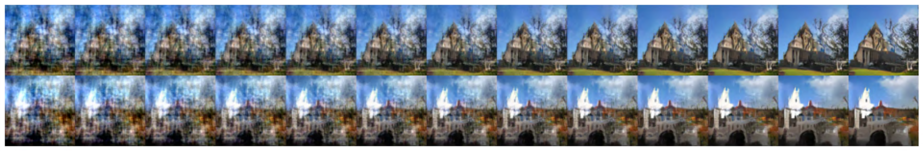


Fig. 5: Illustration of the flow process for LSUN Church, starting from a sample from the data-dependent prior on the leftmost column.

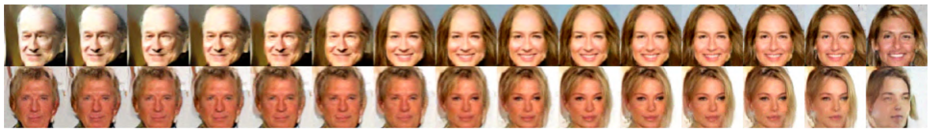


Fig. 6: Interpolation results between leftmost and rightmost samples with CelebA.

by interpolating in the prior space for CelebA in Fig. 6. The model smoothly interpolates in the latent space despite using the DDP, indicating a semantically relevant latent representation that is characteristic of a valid generative model. Finally, to verify that our model has not merely memorized the dataset, we investigate the nearest neighbor samples of the generated images in the training set (see supplementary). The samples produced by FDRL-DDP are distinct from their closest training samples, telling us that FDRL-DDP is capable of generating new and diverse samples beyond the training data.

6.3 Ablations

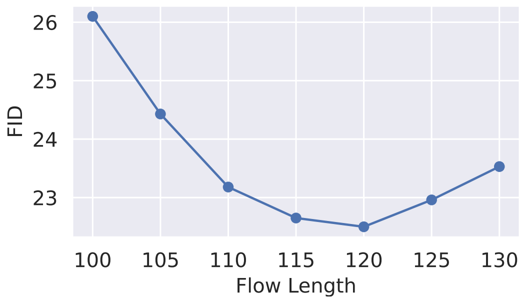


Fig. 7: FID as a function of the total flow length on CIFAR10 for LSIF-χ^2 DDP when sampling from a model trained with $K = 100$.

Data-Dependent Prior vs Uniform Prior We motivate the use of the DDP in conjunction with FDRL by conducting ablations with q' being a uniform prior (UP), $\mathbf{x}_0 \sim U[-1, 1]$. We include qualitative samples of FDRL-UP in the supplementary. Table 2 reports the quantitative scores. Visually, FDRL-UP produces diverse and appealing samples even with a uniform prior. However, the quantitative scores are significantly poorer than using the DDP, which validates our hypothesis in Sec. 3.2.

Length of Flow vs Image Quality Fig. 7 illustrates the relationship between flow length and FID. Although the model is trained with $K = 100$, sampling with K steps at testing does not produce the best image quality, as sampling with an additional $\kappa = 20$ steps yields the best FID score. This supports the two-stage sampling approach discussed in Sec. 4.1.

6.4 Conditional Image Generation

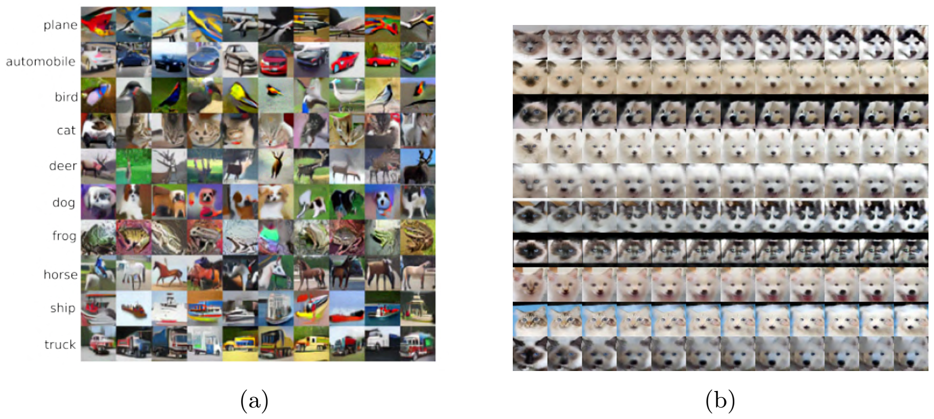


Fig. 8: (a) Class-conditional samples obtained by composing an unconditional FDRL with a pretrained robust classifier. (b) Image-to-image translation process from cat to dog images using FDRL.

We can sample from target densities different from what FDRL was trained on by simply composing density ratios. Consider a multiclass classifier which classifies a given image into one of N classes. We show in the supplementary that we can express such classifiers as a density ratio $p(y = n|\mathbf{x}) = N^{-1}p(\mathbf{x}|y = n)/p(\mathbf{x})$. Thus, we can obtain a conditional density ratio estimator $r_\theta(\mathbf{x}|y = n) = q_T(\mathbf{x})/p(\mathbf{x}|y = n)$ by composing our unconditional estimator $r_\theta(\mathbf{x})$ with the classifier output:

$$r_\theta(\mathbf{x}|y = n) = \frac{1}{N} r_\theta(\mathbf{x}) p(y = n|\mathbf{x})^{-1}. \tag{12}$$

When $r_\theta(\mathbf{x}|y = n)$ is used in the flow in Eq. 9, we perform class-conditional generation. This is conceptually similar to the idea proposed in [33,5], where an unconditional score model $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t))$ is composed with a time-dependent classifier $\nabla_{\mathbf{x}} \log p_t(y|\mathbf{x}(t))$ to form a class-conditional model. However, whereas [33] requires separately training a time-dependent classifier, our formulation allows the use pretrained classifiers off-the-shelf, with *no further retraining*. Inspired by earlier work on image synthesis with robust classifiers [30], we found that using a pretrained adversarially-robust classifier was necessary in obtaining useful gradients for generation. We show our results in Fig. 8a, where each row represents conditional samples of each class in the CIFAR10 dataset.

6.5 Unpaired Image-to-image Translation

FDRL can also be seamlessly applied to unpaired image-to-image-translation (I2I). We simply fix $q'(\mathbf{x})$ to a source image domain and $p(\mathbf{x})$ to a target domain. We then train the model in exactly the same manner as unconditional generation using Algorithm 1.

The I2I model is tested on the Cat2dog dataset [18]. From Fig. 8b, FDRL is able to smoothly translate images of cats to dogs while preserving relevant features, such as background colors, pose and facial tones. For instance, a cat with light fur is translated to a dog with light fur. Quantitatively, the I2I baseline CycleGAN [40] achieves better FID scores than FDRL (see supplementary). However, like many I2I methods [18,4,39], CycleGAN relies on specific inductive biases, such as dual generators and discriminators, and cycle-consistency loss. Future work could explore incorporating these biases into FDRL to improve translation performance.

7 Conclusion

We propose FDRL, a method that enables the stale approximation of the gradient flow to be applied to generative modeling of high-dimensional images. Beyond unconditional image generation, the FDRL framework can be seamlessly adapted to other key applications, including class-conditional generation and unpaired image-to-image translation. Future work could focus on theoretically characterizing the exact distribution induced by the stale SDE of Eq. 5 and further improving our understanding of its convergence properties.

Acknowledgments. This research is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-016).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Altekrüger, F., Hertrich, J., Steidl, G.: Neural wasserstein gradient flows for maximum mean discrepancies with riesz kernels. arXiv preprint arXiv:2301.11624 (2023)
2. Ansari, A.F., Ang, M.L., Soh, H.: Refining deep generative models via discriminator gradient flow. In: ICLR (2021)
3. Arbel, M., Korba, A., Salim, A., Gretton, A.: Maximum mean discrepancy gradient flow. Advances in Neural Information Processing Systems **32** (2019)
4. Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8188–8197 (2020)
5. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in Neural Information Processing Systems **34**, 8780–8794 (2021)
6. Du, C., Li, T., Pang, T., Yan, S., Lin, M.: Nonparametric generative modeling with conditional and locally-connected sliced-wasserstein flows. arXiv preprint arXiv:2305.02164 (2023)
7. Du, Y., Mordatch, I.: Implicit generation and generalization in energy-based models. arXiv preprint arXiv:1903.08689 (2019)
8. Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D.: Robustness (python library) (2019), `https://github.com/MadryLab/robustness`
9. Fan, J., Taghvaei, A., Chen, Y.: Variational wasserstein gradient flow. arXiv preprint arXiv:2112.02424 (2021)
10. Franceschi, J.Y., Gartrell, M., Dos Santos, L., Issenhuth, T., de Bézenac, E., Chen, M., Rakotomamonjy, A.: Unifying gans and score-based diffusion as generative particle models. Advances in Neural Information Processing Systems **36** (2024)
11. Gao, Y., Huang, J., Jiao, Y., Liu, J., Lu, X., Yang, Z.: Deep generative learning via euler particle transport. In: Mathematical and Scientific Machine Learning. pp. 336–368. PMLR (2022)
12. Gao, Y., Jiao, Y., Wang, Y., Wang, Y., Yang, C., Zhang, S.: Deep generative learning via variational gradient flow. In: International Conference on Machine Learning. pp. 2093–2101. PMLR (2019)
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
14. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. Advances in neural information processing systems **30** (2017)
15. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems **33**, 6840–6851 (2020)
16. Jordan, R., Kinderlehrer, D., Otto, F.: The variational formulation of the fokker–planck equation. SIAM journal on mathematical analysis **29**(1), 1–17 (1998)
17. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. Predicting structured data **1**(0) (2006)
18. Lee, H.Y., Tseng, H.Y., Huang, J.B., Singh, M., Yang, M.H.: Diverse image-to-image translation via disentangled representations. In: Proceedings of the European conference on computer vision (ECCV). pp. 35–51 (2018)
19. Liu, C., Zhuo, J., Zhu, J.: Understanding mcmc dynamics as flows on the wasserstein space. In: International Conference on Machine Learning. pp. 4093–4103. PMLR (2019)

20. Liutkus, A., Simsekli, U., Majewski, S., Durmus, A., Stöter, F.R.: Sliced-wasserstein flows: Nonparametric generative modeling via optimal transport and diffusions. In: International Conference on Machine Learning. pp. 4104–4113. PMLR (2019)
21. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957 (2018)
22. Mokrov, P., Korotin, A., Li, L., Genevay, A., Solomon, J.M., Burnaev, E.: Large-scale wasserstein gradient flows. Advances in Neural Information Processing Systems **34**, 15243–15256 (2021)
23. Mroueh, Y., Nguyen, T.: On the convergence of gradient descent in gans: Mmd gan as a gradient flow. In: International Conference on Artificial Intelligence and Statistics. pp. 1720–1728. PMLR (2021)
24. Mroueh, Y., Sercu, T., Raj, A.: Sobolev descent. In: The 22nd International Conference on Artificial Intelligence and Statistics. pp. 2976–2985. PMLR (2019)
25. Nijkamp, E., Hill, M., Zhu, S.C., Wu, Y.N.: Learning non-convergent non-persistent short-run mcmc toward energy-based model. Advances in Neural Information Processing Systems **32** (2019)
26. Pang, B., Han, T., Nijkamp, E., Zhu, S.C., Wu, Y.N.: Learning latent space energy-based prior model. Advances in Neural Information Processing Systems **33**, 21994–22008 (2020)
27. Rhodes, B., Xu, K., Gutmann, M.U.: Telescoping density-ratio estimation. Advances in neural information processing systems **33**, 4905–4916 (2020)
28. Risken, H., Eberly, J.: The fokker-planck equation, methods of solution and applications. Journal of the Optical Society of America B Optical Physics **2**(3), 508 (1985)
29. Santambrogio, F.: {Euclidean, metric, and Wasserstein} gradient flows: an overview. Bulletin of Mathematical Sciences **7**(1), 87–154 (2017)
30. Santurkar, S., Ilyas, A., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Image synthesis with a single (robust) classifier. Advances in Neural Information Processing Systems **32** (2019)
31. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in Neural Information Processing Systems **32** (2019)
32. Song, Y., Ermon, S.: Improved techniques for training score-based generative models. Advances in neural information processing systems **33**, 12438–12448 (2020)
33. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
34. Sugiyama, M., Suzuki, T., Kanamori, T.: Density ratio estimation in machine learning. Cambridge University Press (2012)
35. Sugiyama, M., Suzuki, T., Kanamori, T.: Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. Annals of the Institute of Statistical Mathematics **64**(5), 1009–1044 (2012)
36. Xiao, Z., Kreis, K., Kautz, J., Vahdat, A.: Vaebm: A symbiosis between variational autoencoders and energy-based models. arXiv preprint arXiv:2010.00654 (2020)
37. Xie, J., Lu, Y., Gao, R., Wu, Y.N.: Cooperative learning of energy-based model and latent variable model via mcmc teaching. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
38. Xie, J., Lu, Y., Zhu, S.C., Wu, Y.: A theory of generative convnet. In: International Conference on Machine Learning. pp. 2635–2644. PMLR (2016)

39. Zhao, Y., Chen, C.: Unpaired image-to-image translation via latent energy transport. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16418–16427 (2021)
40. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2223–2232 (2017)

Supplementary Material for "Generative Modeling with Flow-Guided Density Ratio Learning"

A Proofs

Lemma 1. *Let $\gamma = 1$ and assume that $q(\mathbf{x}) \sim U[a, b]$, where a, b are chosen appropriately (e.g., [-1,1] for image pixels). Then the stationary distribution of Eq. 5, $\rho_\infty(\mathbf{x})$, has the same maximum likelihood estimate as $p(\mathbf{x})$, $\arg\max_{\mathbf{x}} \rho_\infty(\mathbf{x}) = \arg\max_{\mathbf{x}} p(\mathbf{x})$.*

Proof. The proof draws inspiration from the Langevin equation. Therefore, let us first consider standard overdamped Langevin SDE:

$$d\mathbf{x}_t = -\nabla f(\mathbf{x}_t)dt + \sqrt{2}d\mathbf{w}_t \tag{13}$$

where f is Lipschitz. The associated Fokker-Planck equation is given by [28]

$$\partial_t \rho_t = \nabla \cdot (\rho \nabla f) + \Delta\rho. \tag{14}$$

where Δ is the Laplace operator. It is straightforward to show that the stationary distribution of Eq. 13 is given by

$$\rho_\infty(\mathbf{x}) = \frac{e^{-f(\mathbf{x})}}{\int e^{-f(\mathbf{x})}d\mathbf{x}} = \frac{e^{-f(\mathbf{x})}}{Z}. \tag{15}$$

We simply verify by substituting Eq. 15 into Eq. 14:

$$\begin{aligned}
\partial_t \rho_\infty &= \nabla \cdot \left(\frac{e^{-f(\mathbf{x})}}{Z}\nabla f\right) + \nabla \cdot \nabla \left(\frac{e^{-f(\mathbf{x})}}{Z}\right) \\
&= \nabla \cdot \left(\frac{e^{-f(\mathbf{x})}}{Z}\nabla f\right) - \nabla \cdot \left(\frac{e^{-f(\mathbf{x})}}{Z}\nabla f\right) \\
&= 0.
\end{aligned}$$

Now, let us return to the stale estimate SDE of Eq. 5 with $\gamma = 1$ and $q(\mathbf{x}) \sim U[a, b] = \frac{1}{b-a} = C$:

$$d\mathbf{x}_t = -\nabla f'(C/p(\mathbf{x}_t))dt + \sqrt{2}d\mathbf{w}_t. \tag{16}$$

Since f is convex and twice-differentiable, f' is a monotonically increasing function. Using what we showed above, the stationary distribution of Eq. 16 is thus

$$\rho_\infty(\mathbf{x}) = \frac{e^{-f'(C/p(\mathbf{x}))}}{\int e^{-f(C/p(\mathbf{x}))}d\mathbf{x}} = \frac{e^{-f'(C/p(\mathbf{x}))}}{Z}. \tag{17}$$

Then we can show that

$$\begin{aligned}
&\arg\max_{\mathbf{x}} \rho_\infty(\mathbf{x}) \\
&= \arg\max_{\mathbf{x}} \frac{e^{-f'(C/p(\mathbf{x}))}}{Z} \\
&= \arg\max_{\mathbf{x}} e^{-f'(C/p(\mathbf{x}))} \\
&= \arg\max_{\mathbf{x}} -f'(C/p(\mathbf{x})) && (e \text{ is monotonically increasing}) \\
&= \arg\min_{\mathbf{x}} f'(C/p(\mathbf{x})) \\
&= \arg\min_{\mathbf{x}} C/p(\mathbf{x}) && (f' \text{ is monotonically increasing}) \\
&= \arg\max_{\mathbf{x}} p(\mathbf{x}) && (p(\mathbf{x}) > 0).
\end{aligned}$$

B Bregman Divergence and *f*-divergence Pairing

We elucidate the full forms of the Bregman divergences here. The LSIF objective $g(y) = \frac{1}{2}(y-1)^2$ is given as

$$\mathcal{L}_{LSIF}(\theta) = \frac{1}{2}\widehat{\mathbb{E}}_p[r_\theta(\mathbf{x})]^2 - \widehat{\mathbb{E}}_{q_\tau}[r_\theta(\mathbf{x})], \tag{18}$$

while the LR objective $g(t) = y\log y - (1+y)\log(1+y)$ is given as

$$\mathcal{L}_{LR}(\theta) = -\widehat{\mathbb{E}}_p\left[\log\frac{1}{1+r_\theta(\mathbf{x})}\right] - \widehat{\mathbb{E}}_{q_\tau}\left[\log\frac{r_\theta(\mathbf{x})}{1+r_\theta(\mathbf{x})}\right]. \tag{19}$$

When computing the LR objective, we find that we run into numerical stability issues when letting $r_\theta(\mathbf{x})$ be the output of an unconstrained neural network and subsequently taking the required logarithms in Eq. 19. To circumvent this issue, we let $r_\theta(\mathbf{x})$ be expressed as the exponential of the neural network's output, i.e., the output of the neural network is $\log r_\theta(\mathbf{x})$. This formulation naturally lends itself to the flow of the KL, JS and logD divergences, whose first derivatives f' that is required in Eq. 9 are also logarithmic functions of $r_\theta(\mathbf{x})$, as seen from Table. 1. We can thus utilize numerically stable routines in existing deep learning frameworks, avoiding the need for potentially unstable operations like exponentiations (see Sec. C for details). As such, we pair LR with the aforementioned divergences and abbreviate the combinations as LR-KL, LR-JS, LR-logD. We did not run into such stability issues for the LSIF objective (Eq. 18) as the model learns to automatically output a non-negative scalar over the course of training, hence for LSIF we allow the neural network to estimate $r_\theta(\mathbf{x})$ directly and pair it with the Pearson-χ^2 divergence. We abbreviate this pairing as LSIF-χ^2.

C Stable Computation of LR and *f*-divergences

As mentioned in Sec. B, computing the logarithm of unconstrained neural networks leads to instabilities in the training process. This is a problem when computing the LR objective in Eq. 19 and the various first derivatives of f-divergences. We can circumvent this by letting $r_\theta(\mathbf{x})$ be expressed as the exponential of the neural network and use existing stable numerical routines to avoid intermediate computations that lead to the instabilities (for example, computing logarithms and exponentials directly). Let us express the neural network output as $NN_\theta(\mathbf{x}) \triangleq \log r_\theta(\mathbf{x})$. The LR objective can then be rewritten as

$$\begin{aligned}\mathcal{L}_{LR}(\theta) &= -\widehat{\mathbb{E}}_p\left[\log \frac{1}{1+r_\theta(\mathbf{x})}\right] - \widehat{\mathbb{E}}_{q_\tau}\left[\log \frac{r_\theta(\mathbf{x})}{1+r_\theta(x)}\right] &(20)\\ &= -\widehat{\mathbb{E}}_p\left[\mathtt{LS}(-NN_\theta(x))\right] - \widehat{\mathbb{E}}_{q_\tau}\left[\mathtt{LS}(NN_\theta(x))\right] &(21)\end{aligned}$$

where $\mathtt{LS}(x) = \log \frac{1}{1+\exp(-x)}$, the log-sigmoid function, which has stable implementations in modern deep learning libraries.

Similarly for the f-divergences whose first derivatives involve logarithms, we can calculate them stably as

$$\begin{aligned} f'_{KL}(r_\theta(\mathbf{x})) &= \log r_\theta(\mathbf{x}) + 1 = NN_\theta(\mathbf{x}) + 1 &(22)\\ f'_{JS}(r_\theta(\mathbf{x})) &= \log \frac{2r_\theta(\mathbf{x})}{1+r_\theta(\mathbf{x})} = \log 2 + \mathtt{LS}(NN_\theta(\mathbf{x})) &(23)\\ f'_{logD}(r_\theta(\mathbf{x})) &= \log(r_\theta(\mathbf{x})+1) + 1 = -\mathtt{LS}(-NN_\theta(\mathbf{x})) + 1. &(24)\end{aligned}$$

D Classifiers are Density Ratio Estimators

To perform class-conditional generation in the FDRL framework, we would like to estimate the density ratio of a certain class over the data distribution: $p(\mathbf{x}|y = n)/p(\mathbf{x})$. With Bayes rule, we can write this as

$$\begin{aligned}\frac{p(\mathbf{x}|y=n)}{p(\mathbf{x})} &= \frac{p(y=n|\mathbf{x})p(\mathbf{x})/p(y=n)}{p(\mathbf{x})} &(25)\\ &= \frac{p(y=n|\mathbf{x})}{p(y=n)}. &(26)\end{aligned}$$

The denominator term $p(y = n)$ can be viewed as a constant, e.g. assume the N classes are equally distributed, then $p(y = n) = 1/N$. Therefore, we have that the class probability given by the softmax output of a classifier is actually a density ratio:

$$Np(y=n|\mathbf{x}) = \frac{p(\mathbf{x}|y=n)}{p(\mathbf{x})}. \quad (27)$$

We can use this equation to convert an unconditional FDRL to a class-conditional generator. Recall the stale approximation of the gradient flow:

$$d\mathbf{x}_t = -\nabla_{\mathbf{x}} f'(r_\theta(\mathbf{x}_t))dt + \sqrt{2\gamma}d\mathbf{w}_t \quad (28)$$

Consider the case where we have a trained unconditional FDRL estimator r_θ. We can multiply the inverse of the classifier output with $r_\theta(\mathbf{x}_t) = \tilde{q}_\tau(\mathbf{x}_t)/p(\mathbf{x}_t)$ to get a density ratio between $\tilde{q}_\tau(\mathbf{x}_t)$ and the conditional data distribution $p(\mathbf{x}_t|y=n)$:

$$\begin{aligned} r_\theta(\mathbf{x}_t)p(y=n|\mathbf{x}_t)^{-1} &= \frac{\tilde{q}_\tau(\mathbf{x}_t)}{p(\mathbf{x}_t)} \frac{Np(\mathbf{x}_t)}{p(\mathbf{x}_t|y=n)} \\ &= N\frac{\tilde{q}_\tau(\mathbf{x}_t)}{p(\mathbf{x}_t|y=n)}. \end{aligned}$$

That is, we took our unconditional model and converted it to a conditional generative model by composing it with a pretrained classifier. To get the correct class-conditional density ratio that can be used for generation, we should therefore compute

$$r_\theta(\mathbf{x}_t|y=n) = \frac{1}{N} r_\theta(\mathbf{x}_t)p(y=n|\mathbf{x}_t)^{-1} \tag{29}$$

and use this conditional density ratio estimator in our sampling method of Algorithm Eq. 2.

Table 3: FID scores for image-to-image translation with the Cat2dog dataset.

Model	FID ↓
FDRL	108.10
CycleGAN	51.79

E Experimental Details

Table 4: Network structures for the density ratio estimator $r_\theta(\mathbf{x})$.

CIFAR10	CelebA 64	LSUN Church 128	Cat2dog 64
3×3 Conv2d, 128	3×3 Conv2d, 64	3×3 Conv2d, 64	3×3 Conv2d, 64
3 × ResBlock 128	ResBlock Down 64	ResBlock Down 64	ResBlock Down 64
ResBlock Down 256	ResBlock Down 128	ResBlock Down 128	ResBlock Down 128
2 × ResBlock 256	ResBlock 128	ResBlock Down 128	Self Attention 128
ResBlock Down 256	ResBlock Down 256	ResBlock 128	ResBlock Down 128
2 × ResBlock 256	ResBlock 256	ResBlock Down 256	ResBlock Down 256
ResBlock Down 256	ResBlock Down 256	ResBlock 256	Global Mean Pooling
2 × ResBlock 256	ResBlock 256	ResBlock Down 256	Dense → 1
Global Mean Pooling	Global Mean Pooling	ResBlock 256	
Dense → 1	Dense → 1	Global Mean Pooling	
		Dense → 1	

E.1 Unconditional Image Generation.

For all datasets, we perform random horizontal flip as a form of data augmentation and normalize pixel values to the range [-1, 1]. For CelebA, we center crop the image to 140×140 before resizing to 64×64. For LSUN Church, we resize the image to 128×128 directly. We use the same training hyperparameters across all three datasets, which is as follows. We train all models with 100000 training steps with the Adam optimizer with a batch size of 64, except for CIFAR10 with uniform prior where we trained for 120000 steps. We use a learning rate of 1×10^{-4} and decay twice by a factor of 0.1 when there are 20000 and 10000 remaining steps. We set the number of flow steps to $K = 100$ at training time. When sampling, we set $\kappa = 20$ for all DDP experiments and $\kappa = 10$ for UP experiments. Other flow hyperparameters include a step size of $\eta = 3$ and noise factor $\nu = 10^{-2}$. The specific ResNet architectures are given in Table 4. We update model weights using an exponential moving average [32] given by $\boldsymbol{\theta}' \leftarrow m\boldsymbol{\theta}' + (1 - m)\boldsymbol{\theta}_i$, where $\boldsymbol{\theta}_i$ is the parameters of the model at the i-th training step, and $\boldsymbol{\theta}'$ is an independent copy of the parameters that we save and use for evaluation. We set $m = 0.998$. We use the LeakyReLU activation for our networks with a negative slope of 0.2. We experimented with spectral normalization and self attention layers for unconditional image generation, but found that training was stable enough such that they were not worth the added computational cost. The FID results in the main text are obtained by generating 50000 images, and testing the results against the training set for both CIFAR10 and CelebA.

E.2 Conditional Generation with Robust Classifier.

The unconditional model used for conditional generation is the same model obtained from the section above. The pretrained robust classifier checkpoint is obtained from the `robustness` Python library [8]. It is based on a ResNet50 architecture and is trained with L_2-norm perturbations of $\varepsilon = 1$.

We choose the LR-KL variant for our results in Fig. 8 of the main text. This means that our conditional flow is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - 2\eta\nabla_{\mathbf{x}} \log\left(r_\theta(\mathbf{x}_k) * \frac{1}{N} p(y = n|\mathbf{x}_k)^{-1}\right) + \nu\boldsymbol{\xi}_k \tag{30}$$

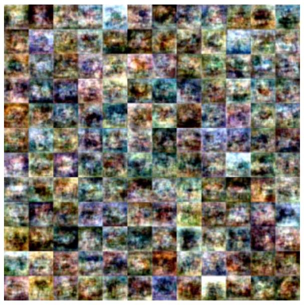
$$= \mathbf{x}_k - 2\eta\nabla_{\mathbf{x}} \left(\log r_\theta(\mathbf{x}_k) - \phi \log p(y = n|\mathbf{x}_k)\right) + \nu\boldsymbol{\xi}_k \tag{31}$$

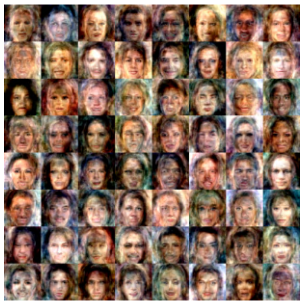
where in the second line we introduce ϕ as a parameter that scales the magnitude of the classifier's gradients so they are comparable to the magnitude of FDRL's gradients. We use $\phi = 0.1$.

E.3 Unpaired Image-to-image Translation.

The Cat2dog dataset contains 871 Birman cat images and 1364 Samoyed and Husky dog images. 100 of each are set aside as test images. We first resize the images to 84×84 before center cropping to 64×64. Due to the relatively small size of the dataset, we use a shallower ResNet architecture as compared to CelebA 64^2 despite the same resolution (Table 4) to prevent overfitting. We also utilize spectral normalization and a self attention layer at the 128-channel level to further boost stability. We set $K = 100$ during training and $K = 110$ at test time. As we observed the model tends to diverge late in training, we limit the number of training steps to 40000, with a decay factor of 0.1 applied to the learning rate at steps 20000 and 30000. All other hyperparameters are kept identical to the experiments on unconditional image generation. We report results for LSIF-χ^2, although we have experimented with LR-KL and found performance to be similar. The FID result in the main text is obtained by translating the 100 test cat images, and testing the results against the 100 test dog images.

F Data-Dependent Prior Samples


(a)


(b)

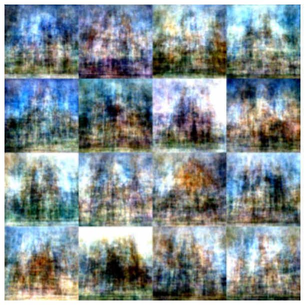
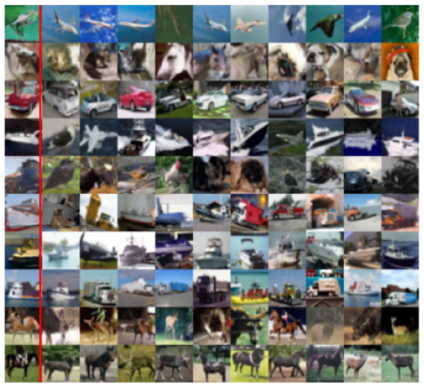
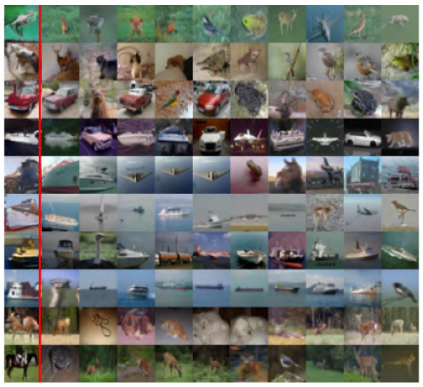

(c)

Fig. 9: Samples drawn from the data-dependent priors of (a) CIFAR10 32^2, (b) CelebA 64^2 and (c) LSUN Church 128^2.

G Nearest Neighbors

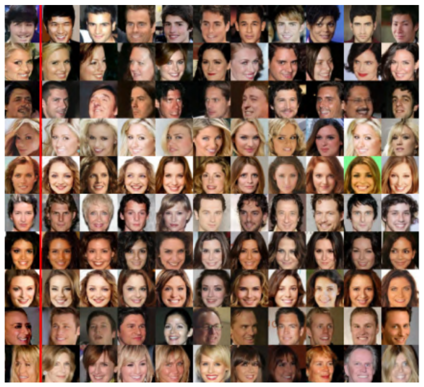


(a)



(b)

Fig. 10: Nearest neighbor images for CIFAR10 as measured by L_2 distance in (a) the feature space of an Inception V3 network pretrained on ImageNet and (b) data space. The column to the left of the red line are samples from FDRL LSIF-χ^2. The images to the right of the line are the 10 nearest neighbors in the training dataset.


(a)

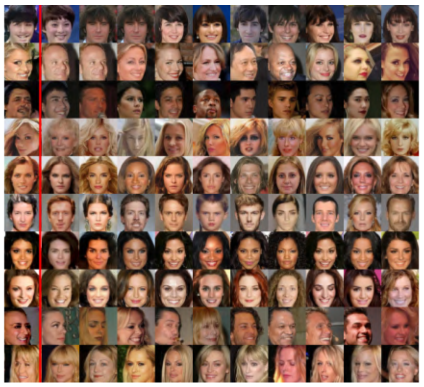

(b)

Fig. 11: Nearest neighbor images for CelebA as measured by L_2 distance in (a) the feature space of an Inception V3 network pretrained on ImageNet and (b) data space. The column to the left of the red line are samples from FDRL LSIF-χ^2. The images to the right of the line are the 10 nearest neighbors in the training dataset.

H Uncurated Samples FDRL-DDP

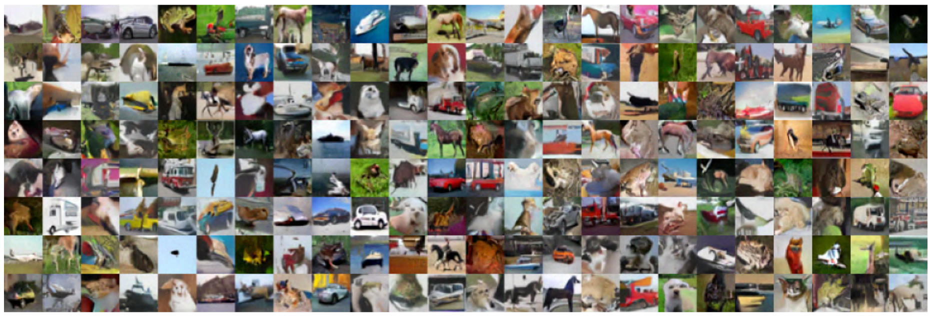


Fig. 12: Uncurated samples of CIFAR10 LSIF-χ^2.

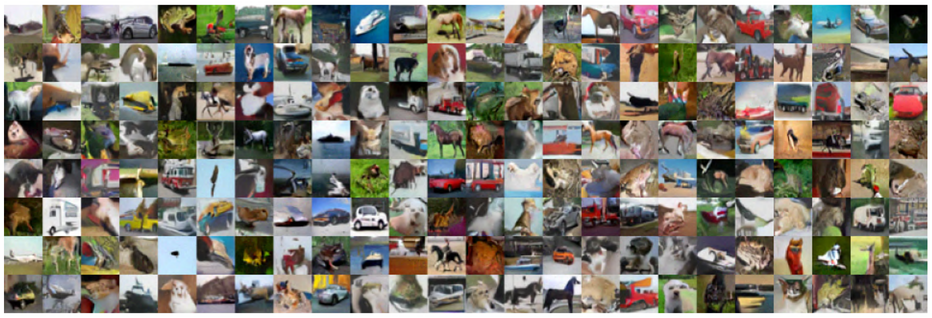


Fig. 13: Uncurated samples of CIFAR10 LR-KL.

Fig. 14: Uncurated samples of CIFAR10 LR-JS.

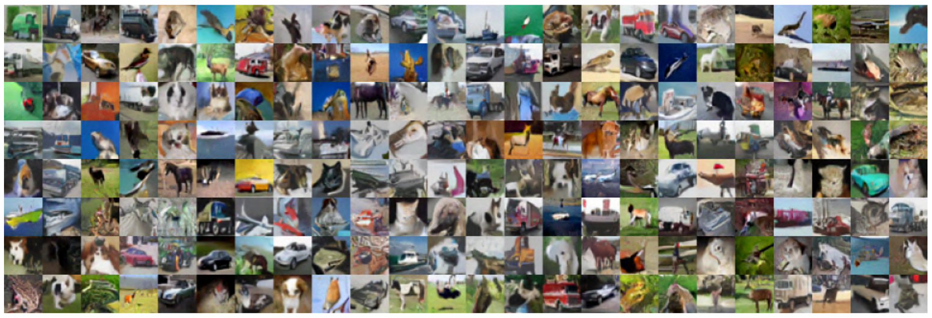


Fig. 15: Uncurated samples of CIFAR10 LR-logD.

Fig. 16: Uncurated samples of CelebA LSIF-χ^2.

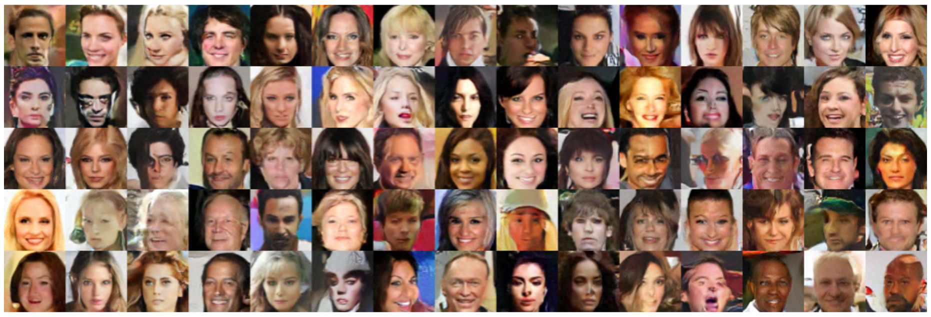


Fig. 17: Uncurated samples of CelebA LR-KL.

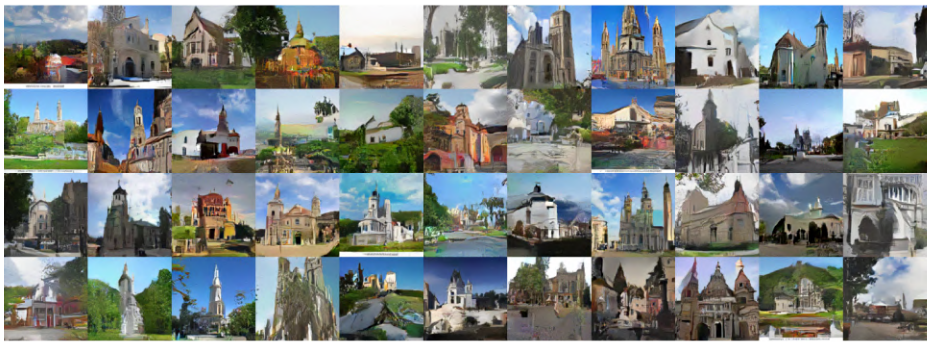
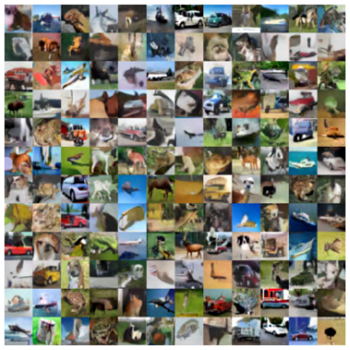


Fig. 18: Uncurated samples of LSUN Church LSIF-χ^2.

I Uncurated Samples FDRL-UP


(a) LSIF-χ^2 uniform prior.

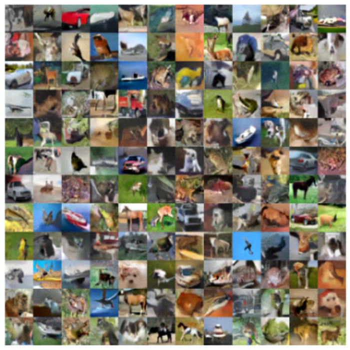

(b) LR-KL uniform prior.

Fig. 19: Uncurated CIFAR10 samples with FDRL-UP.