IMPROVED AUTOREGRESSIVE MODELING WITH DISTRIBUTION SMOOTHING

Anonymous authors
Paper under double-blind review

ABSTRACT

While autoregressive models excel at image compression, their sample quality is often lacking. Inspired by randomized smoothing for adversarial defense, we incorporate randomized smoothing techniques into autoregressive generative modeling. We first model a smoothed version of the data distribution and then recover the data distribution by learning to reverse the smoothing process. We demonstrate empirically on a 1-d dataset that by appropriately choosing the smoothing level, we can keep the proposed process relatively easier to model than directly learning a data distribution with a high Lipschitz constant. Since autoregressive generative modeling consists of a sequence of 1-d density estimation problems, we believe the same arguments can be generalized to an autoregressive model. This seemingly simple procedure drastically improves the sample quality of existing autoregressive models on several synthetic and real-world datasets while obtaining competitive likelihoods on synthetic datasets.

1 INTRODUCTION

Autoregressive models have exhibited promising results in a variety of downstream tasks. For instance, they have shown success in compressing real-world images (Minnen et al., 2018), synthesizing realistic speech (Oord et al., 2016a) and providing competitive game strategies (Vinyals et al., 2019). However, the sample quality of autoregressive models on real-world image datasets is still lacking, which, we believe, can be caused by the following two reasons.

The first one is the manifold hypothesis, which states that many real world data distributions (*e.g.* natural images) may lie in the vicinity of a low-dimensional manifold (Belkin & Niyogi, 2003) and can often have complicated densities with sharp transitions (*i.e.* high Lipschitz constants) that are difficult to model. While addressing the manifold hypothesis can already be a challenging problem even with infinitely many data, in real life, one might only have access to a limited amount of training samples, making learning even harder. Besides, an insufficient amount of training data can also give rise to "compounding error" issues. To see this, we note that an autoregressive model relies on the previous context to make prediction for the next-step; once a mistake is made, the model is likely to make another mistake which compounds (Kääriäinen, 2006), eventually resulting in samples that are spurious but have high-likelihoods when evaluated by the model.

The spurious sample, in some sense, resembles an "adversarial example" (Goodfellow et al., 2014). Inspired by the recent success of randomized smoothing techniques in adversarial defense (Cohen et al., 2019), we propose to apply randomized smoothing to autoregressive generative modeling. More specifically, we propose to address a density estimation problem via a two-stage process. Unlike Cohen et al. (2019) which applies smoothing to the model to make it more robust, we apply smoothing to the data distribution, where a symmetric and stationary distribution is convolved with the data distribution to make the new distribution "smooth". In the first stage, we model the smoothed version of the data distribution using an autoregressive model. In the second stage, we reverse the smoothing process by either applying a gradient-based denoising approach (Saremi et al., 2018) or introducing another autoregressive model to recover the original data distribution. By choosing an appropriate smoothing, we aim to make each step easier than the original learning problem: smoothing facilitates learning in the first stage, and generating a sample given a noisy one is easier than generating one from scratch.

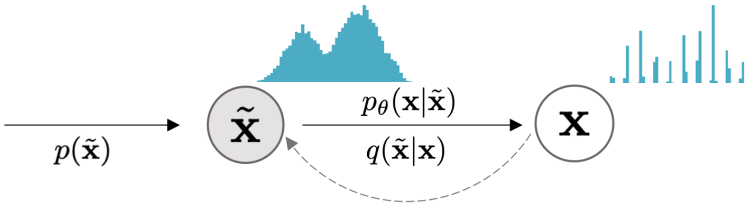


Figure 1: Overview of our method. From a data distribution ($\mathbf{x}$) we inject noise ($q(\tilde{\mathbf{x}}|\mathbf{x})$) which makes the distribution smoother ($\tilde{\mathbf{x}}$); then we model the smoothed distribution ($p_\theta(\tilde{\mathbf{x}})$) as well as the denoising step ($p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$), forming a two-step model.

We show with extensive experimental results that our approach is able to drastically improve the sample quality of current autoregressive models on several synthetic datasets and real-world image datasets, while obtaining competitive likelihoods on synthetic datasets. We empirically demonstrate that our method can also be applied to density estimation, image inpainting and image denoising.

2 Background

Given D-dimensional i.i.d samples $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ from a continuous data distribution $p_{\text{data}}(\mathbf{x})$, we want to perform density estimation on $p_{\text{data}}(\mathbf{x})$ with a density function $p_\theta(\mathbf{x})$ often parameterized by a deep network, where θ are parameters of the model. A commonly used approach for density estimation is maximum likelihood estimation (MLE), i.e., by maximizing $L(\theta) \triangleq \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i)$.

2.1 Autoregressive models

An autoregressive model (Salimans et al., 2017; Germain et al., 2015) decomposes a joint distribution into univariate conditionals:

$$p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|\mathbf{x}_{<i}), \tag{1}$$

where x_i refers to the i-th index of $\mathbf{x}$, and $\mathbf{x}_{<i}$ refers to the components with indices smaller than i in the given ordering. In general, an autoregressive model parameterizes each conditional $p(x_i|\mathbf{x}_{<i})$ using a pre-specified density function (*e.g.* mixture of logistics). However, this approach bounds the capacity of the model by limiting the number of modes for each conditional.

Although autoregressive models have achieved top likelihoods amongst all types of density based models, they are still unable to generate high quality image samples (Du & Mordatch, 2019). We believe this can be caused by the following two reasons.

2.2 Manifold Hypothesis

In many domains where very high-dimensional data is concerned, several existing methods (Roweis & Saul, 2000; Tenenbaum et al., 2000) rely on the manifold hypothesis , i.e. that high-dimensional data tends to lie in a low dimensional manifold (Narayanan & Mitter, 2010). If the manifold hypothesis is true, then the density of the data distribution is not even well defined in the ambient space; if the manifold hypothesis is only approximately correct (the data lies in the vicinity of the manifold), then this suggests that points that are very close to the manifold would have high density, and ones that are slightly further have zero density. Thus we may expect the data density around the manifold to have large first-order derivatives, i.e. the density function has a high Lipschitz constant (if not infinity). For certain likelihood models, such as normalizing flows (Cornish et al., 2019), it has been observed that having high Lipschitz constants from such datasets is detrimental to training; this could also negatively impact learning of autoregressive models.

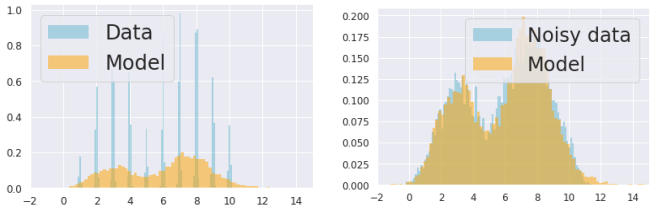
2.3 Compounding errors in autoregressive modeling

Autoregressive models can also be susceptible to compounding errors from the conditional distributions (Lamb et al., 2016). We notice that an autoregressive model $p_\theta(\mathbf{x})$ learns the joint density $p_{\text{data}}(\mathbf{x})$ by matching each of the conditional $p_\theta(x_i|\mathbf{x}_{<i})$ with $p_{\text{data}}(x_i|\mathbf{x}_{<i})$. In reality, we often have

access to very limited amount of training data, which makes it hard for an autoregeressive model to capture all the conditional distributions correctly due to the curse of dimensionality. During sampling time, since a prediction of the current state is based on the values of the previous states, once a mistake is made at a previous step, the model is likely to make more mistakes in the later steps, eventually generating a sample $\hat{\mathbf{x}}$ that is far from being an actual image, but is mistakenly assigned a high density by the model.

The "erroneous" sample $\hat{\mathbf{x}}$, in some sense, resembles an adversarial example, i.e., an input that causes the model to make mistakes. Recent works (Cohen et al., 2019) in adversarial defense have shown that random noise can be used to improve the model's robustness to adversarial perturbations — a process during which adversarial examples that are close to actual data are generated to fool the model. We hypothesize that such approach can also be applied to improve an autoregressive modeling process by making the model less vulnerable to compounding errors occurred during density estimation. Inspired by the success of randomized smoothing in adversarial defense (Cohen et al., 2019), we propose to apply smoothing to autoregressive modeling to address the problems mentioned above.

3 Generative models with distribution smoothing



(a) Data (b) Smoothed data

Figure 2: Visualization of a 1-d data distribution without smoothing (a) or with smoothing (b), modeled by the same mixture of logistics model.

In the following, we propose to decompose a density estimation task into a smoothed data modeling problem and a reverse smoothing problem.

3.1 Randomized smoothing process

Inspired by the success of "randomized smoothing" in adversarial defense (Cohen et al., 2019), we propose to apply randomized smoothing to autoregressive generative modeling. Unlike Cohen et al. (2019) which uses randomized smoothing to smooth a given model, we apply randomized smoothing directly to $p_{\text{data}}(\mathbf{x})$ to make the data distribution smoother. To do this, we introduce a smoothing distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$ — a distribution that is symmetric and stationary (*e.g.* a Gaussian or Laplacian kernel), and convolve it with $p_{\text{data}}(\mathbf{x})$ to obtain a new distribution $q(\tilde{\mathbf{x}}) \triangleq \int q(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})d\mathbf{x}$. When $q(\tilde{\mathbf{x}}|\mathbf{x})$ is a normal distribution, this convolution process is equivalent to perturbing the data distribution with Gaussian noise, which, intuitively, will make the data distribution smoother.

Theorem 1. *Given a continuous 1-d distribution $p(x)$ that is supported on $\mathbb{R}$, for any 1-d distribution $q(\tilde{x}|x)$ that is symmetric (*i.e. $q(\tilde{x}|x) = q(x|\tilde{x})$*), stationary (*i.e. *translation invariant) and satisfies $\lim_{x\to\infty} p(x)q(x|\tilde{x}) = 0$ for any given $\tilde{x}$, we have $Lip(q(\tilde{x})) \leq Lip(p_{data}(x))$, where $q(\tilde{x}) \triangleq \int q(\tilde{x}|x)p(x)dx$ and $Lip(\cdot)$ denotes the Lipschitz constant of the given 1-d function.*

Theorem 1 shows that convolving a 1-d data distribution $p_{\text{data}}(x)$ with certain choices of $q(\tilde{x}|x)$ (*e.g.* $\mathcal{N}(\tilde{x}|x, \sigma^2)$) can reduce the Lipschitzness (*i.e.* increase the smoothness) of $p_{\text{data}}(x)$.

To provide some intuition, we visualize a 1-d example in figure 2a, where we want to model a ten-mode data distribution with a mixture of logistics model. If our model has three logistic components, there is almost no way for the model, which only has three modes, to perfectly fit this data distribution, which has ten separate modes with sharp transitions. The model, after training (see figure 2a), mistakenly assigns a much higher density to the low density regions between nearby modes. If we

convolve the data distribution with $q(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, 0.5^2)$, the new distribution becomes smoother than the original data distribution (see figure 2b) and can be captured reasonably well even by the same mixture of logistics model with only three modes (see figure 2b). Comparing the same model's performance on the two density estimation tasks, we can see that the model is doing a better job at modeling the smoothed version of the data distribution than the original data distribution, which has a high Lipschitz constant.

This smoothing process can also be understood as a regularization term for the original maximum likelihood objective (on the un-smoothed data distribution), encouraging the learned model to be smooth, as formalized by the following theorem:

Theorem 2. *Assume that the symmetric and stationary smoothing distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$ has small variance and negligible higher order moments, then*

$$\mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[p_\theta(\tilde{\mathbf{x}})] \approx \mathbb{E}_{p_{data}(\mathbf{x})}\left[\log p_\theta(\mathbf{x}) + \frac{\eta}{2}\sum_i \frac{\partial^2 \log p_\theta}{\partial x_i^2}\right],$$

for some constant η.

Theorem 2 shows that our smoothing process provides a regularization effect on the original objective $\mathbb{E}_{p_{\text{data}}(\mathbf{x})}[\log p_\theta(\mathbf{x})]$ where no noise is added. This regularization effect can, intuitively, increase the generalization capability of the model. The argument is similar to that used in (Bishop, 1995) except that we consider a more general generative modeling case as opposed to supervised learning with squared error.

3.2 Autoregressive distribution smoothing models

Motivated by the previous 1-d example, instead of directly modeling $p_{\text{data}}(\mathbf{x})$, which can have a high Lipschitz constant, we propose to first train an autoregressive model on the smoothed version of the data distribution $q(\tilde{\mathbf{x}})$. Although the smoothing process makes the distribution easier to learn, it also introduces bias. Thus, we need an extra step to debias the learned distribution by reverting the smoothing process.

If our goal is to generate samples for $p_{\text{data}}(\mathbf{x})$, when $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$ we can directly apply "single-step denoising" as discussed in (Saremi et al., 2018). More specifically, given smoothed samples $\tilde{\mathbf{x}}$ from $p_\theta(\tilde{\mathbf{x}})$, we can denoise samples via:

$$\bar{\mathbf{x}} = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log p_\theta(\tilde{\mathbf{x}}), \tag{2}$$

which only requires the knowledge of $p_\theta(\tilde{\mathbf{x}})$ and samples from it. However, this approach does not provide a likelihood estimation of $p_{\text{data}}(\mathbf{x})$ and Eq. (2) only works when $q(\tilde{\mathbf{x}}|\mathbf{x})$ is Gaussian (though alternative denoising updates for other smoothing processes could be derived under the Empirical Bayes framework (Raphan & Simoncelli, 2011)).

To invert more general smoothing distributions (beyond Gaussians) as well as obtaining the likelihood estimation of $p_{\text{data}}(\mathbf{x})$, we introduce a second autoregressive model $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$. The parameterized joint density $p_\theta(\mathbf{x}, \tilde{\mathbf{x}})$ can then be computed as $p_\theta(\mathbf{x}, \tilde{\mathbf{x}}) = p_\theta(\mathbf{x}|\tilde{\mathbf{x}})p_\theta(\tilde{\mathbf{x}})$. To obtain our approximation of $p_{\text{data}}(\mathbf{x})$, we need to integrate over $\tilde{\mathbf{x}}$ on the joint distribution $p_\theta(\mathbf{x}, \tilde{\mathbf{x}})$ to obtain $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \tilde{\mathbf{x}}) d\tilde{\mathbf{x}}$, which is in general intractable. However, we can easily obtain an evidence lower bound (ELBO):

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log p_\theta(\tilde{\mathbf{x}})] - \mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log q(\tilde{\mathbf{x}}|\mathbf{x})] + \mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\tilde{\mathbf{x}})]\right]. \tag{3}$$

Note that when $q(\tilde{\mathbf{x}}|\mathbf{x})$ is fixed, the entropy term $\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log q(\tilde{\mathbf{x}}|\mathbf{x})]$ is a constant with respect to the optimization parameters. Maximizing the ELBO is then equivalent to maximizing:

$$J(\theta) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log p_\theta(\tilde{\mathbf{x}})] + \mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\tilde{\mathbf{x}})]\right]. \tag{4}$$

From equation 4, we can see that optimizing $J(\theta)$ is equivalent to optimizing the two models $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ separately via maximum likelihood estimation.

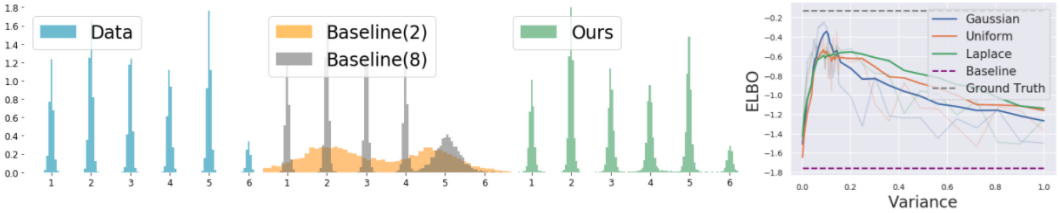


Figure 3: Density estimation on 1-d synthetic dataset. In the second figure, the digit in the parenthesis denotes the number of mixture components used in the baseline mixture of logistics model. In comparison, our model in the third figure uses only 2 mixture of logistics components for each univariate conditional distribution.

3.3 Tradeoff in modeling

In general, there is a trade-off between the difficulty of modeling $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$. To see this, let us consider two extreme cases for the variance of $q(\tilde{\mathbf{x}}|\mathbf{x})$ — when $q(\tilde{\mathbf{x}}|\mathbf{x})$ has a zero variance and an infinite variance. In the first case, when $q(\tilde{\mathbf{x}}|\mathbf{x})$ has a zero variance, $q(\tilde{\mathbf{x}}|\mathbf{x})$ is simply zero, meaning that no noise is added to the data distribution. In this case, modeling the new distribution would be equivalent to modeling $p_{\text{data}}(\mathbf{x})$, which can be hard as discussed above. The reverse smoothing process, however, would be easy since $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ can simply be an identity map to perfectly invert the smoothing process. In the second case when $q(\tilde{\mathbf{x}}|\mathbf{x})$ has an infinite variance, modeling $p(\tilde{\mathbf{x}})$ would be easy because all the information about the original data is lost, and $p(\tilde{\mathbf{x}})$ would be close to the smoothing distribution. Modeling $p(\mathbf{x}|\tilde{\mathbf{x}})$, on the other hand, is equivalent to directly modeling $p_{\text{data}}(\mathbf{x})$, which can be challenging.

Thus, the key here is to appropriately choose a smoothing level so that both $q(\tilde{\mathbf{x}})$ and $p(\mathbf{x}|\tilde{\mathbf{x}})$ can be approximated relative well by existing autoregressive models. In general, the optimal variance might be hard to find. Although one can train $q(\tilde{\mathbf{x}}|\mathbf{x})$ by jointly optimizing ELBO, in practice, we find this approach often assigns a very large variance to $q(\tilde{\mathbf{x}}|\mathbf{x})$, which can trade-off sample quality for better likelihoods on high dimensional image datasets. We find empirically that a pre-specified $q(\tilde{\mathbf{x}}|\mathbf{x})$ chosen by heuristics (Saremi & Hyvarinen, 2019; Garreau et al., 2017) is able to generate much better samples than training $q(\tilde{\mathbf{x}}|\mathbf{x})$ via ELBO. In this paper, we will focus on the sample quality and leave the training of $q(\tilde{\mathbf{x}}|\mathbf{x})$ for future work.

4 Experiments

In this section, we demonstrate empirically that by appropriately choosing the smoothness level of randomized smoothing, our approach is able to drastically improve the sample quality of existing autoregressive models on several synthetic and real-world datasets while retaining competitive likelihoods on synthetic datasets.

4.1 Choosing the smoothing distribution

To help us build insights into the selection of the smoothing distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$, we first focus on a 1-d multi-modal distribution (see figure 3 leftmost panel). We start with the exploration of three different types of smoothing distribution – Gaussian distribution, Laplace distribution, and uniform distribution. For each type of distribution, we perform a grid search to find the optimal variance. Since our approach requires the modeling of both $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$, we stack $\tilde{\mathbf{x}}$ and $\mathbf{x}$ together, and use a MADE model (Germain et al., 2015) with a mixture of two logistic components to parameterize $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ at the same time. For the baseline model, we train a mixture of logistics model directly on $p_{\text{data}}(\mathbf{x})$. We compare the results in the middle two panels in figure 3.

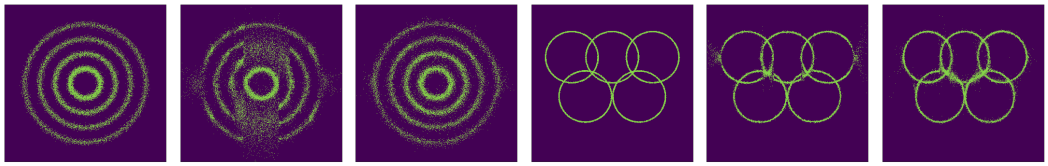
We find that although the baseline with eight logistic components has the capacity to perfectly model the multi-modal data distribution, which has six modes, the baseline model still fails to do so. We believe this can be caused by optimization or initialization issues for modeling a distribution with a high Lipschitz constant. Our method, on the other hand, demonstrates more robustness by successfully modeling the different modes in the data distribution even when using only two mixture components for both $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$.

Table 1: Negative log-likelihoods on 2-d synthetic datasets (lower is better). We compare with MADE (Germain et al., 2015), RealNVP (Dinh et al., 2016), CIF-RealNVP (Cornish et al., 2019).

Dataset	RealNVP	CIF-RealNVP	MADE (2 mixtures)	MADE (5 mixtures)	Ours (2 mixtures)
Rings	2.81	2.81	3.38	2.99	**2.77**
Olympics	1.80	1.74	2.18	**0.83**	0.99

For all the three types of smoothing distributions, we observe a reverse U-shape correlation between the variance of $q(\tilde{\mathbf{x}}|\mathbf{x})$ and ELBO — with ELBO first increasing as the variance increases and then decreasing as the variance reaches beyond a certain point. The results match our discussion on the trade-off between modeling $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ in Section 3.3. We notice from the empirical results that Gaussian smoothing is able to obtain better ELBO than the other two distributions. Thus, we will use $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$ for the later experiments.

4.2 2-D SYNTHETIC DATASETS



(a) Rings (b) MADE (5) (c) Ours (d) Olympics (e) MADE (5) (f) Ours

Figure 4: Samples on 2-d synthetic datasets. We use a MADE model with comparable number of parameters for both our method and the baseline. Our model uses 2 mixture of logistics, while the baseline uses 5 (more) mixture of logistics.

In this section, we consider two challenging 2-d multi-modal synthetic datasets (see figure 4). We use a MADE model with comparable number of total parameters for both the baseline and our approach. For the baseline, we train the MADE model directly on the data. For our randomized smoothing model, we choose $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, 0.3^2 I)$ to be the smoothing distribution. We observe that with this randomized smoothing approach, our model is able to generate better samples than the baseline (according to a human observer) even when using less logistic components (see figure 4). We also provide the negative log-likelihoods in Tab. 1.

4.3 IMAGE EXPERIMENTS

In this section, we focus on three common image datasets, namely MNIST, CIFAR-10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015). We select $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$ to be the smoothing distribution. We use PixelCNN++ (Salimans et al., 2017) as the model architecture for both $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$. We provide more details about settings in Appendix B.

Image generation. For image datasets, we select the σ of $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$ according to analysis in (Saremi & Hyvarinen, 2019) (see Appendix B for more details). Since $q(\tilde{\mathbf{x}}|\mathbf{x})$ is a Gaussian distribution, we can apply "single-step denoising" to reverse the smoothing process for samples drawn from $p_\theta(\tilde{\mathbf{x}})$. In this case, the model $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ is not required for sampling since the gradient of $p_\theta(\tilde{\mathbf{x}})$ can be used to reverse the smoothed samples drawn from itself (*i.e.* $p_\theta(\tilde{\mathbf{x}})$) (see equation 2). We present smoothed samples from $p_\theta(\tilde{\mathbf{x}})$, reversed smoothing samples processed by "single-step denoising" and processed by $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ in figure 5. For comparison, we also present samples from a PixelCNN++ with parameters comparable to the sum of total parameters of $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$. We find that by using this randomized smoothing approach, we are able to drastically improve the sample quality of PixelCNN++ (see the rightmost panel in figure 5).

We note that with only $p_\theta(\tilde{\mathbf{x}})$, a PixelCNN++ optimized on the smoothed data, we are already able to obtain more image-like samples compared to the original PixelCNN++ method. However, $p_\theta(\tilde{\mathbf{x}})$ alone does not provide likelihood estimation of $p_{\text{data}}(\mathbf{x})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ is required for the computation

Figure 5: From left to right: samples from $p_\theta(\tilde{\mathbf{x}})$, "single-step denoising" samples from $p_\theta(\tilde{\mathbf{x}})$, samples from $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$, and samples from the baseline PixelCNN++ model. The second column ("single-step denoising") has "smoother" samples compared to the third column (modeled by another PixelCNN++) . All the samples are not conditioned on class labels.

of likelihoods. We report the sample quality evaluated by FID and Inception scores in Tab. 2. We provide more image samples in Appendix B.2 and nearest neighbors analysis in Appendix B.3.

Although our method is able to obtain better samples compared to the original PixelCNN++, our model has worse likelihoods. We believe this is because likelihood and sample quality are not always directly correlated as discussed in Theis et al. (2015). We also tried training the variance for $q(\tilde{\mathbf{x}}|\mathbf{x})$ by jointly optimizing ELBO. Although training the variance can produce better likelihoods, it does not generate samples with comparable quality as our method (*i.e.* choosing variance by heuristics). Thus, it is hard to conclusively determine what is the best way of choosing $q(\tilde{\mathbf{x}}|\mathbf{x})$.

Denoising. We notice that the reverse smoothing process can also be understood as a denoising process. Besides the "single-step denoising" approach shown above, we can also apply $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ to denoise images. To visualize the denoising performance, we sample $\mathbf{x}_{\text{test}}$ from the test set and perturb $\mathbf{x}_{\text{test}}$ with $q(\tilde{\mathbf{x}}|\mathbf{x})$ to obtain a noisy sample $\tilde{\mathbf{x}}_{\text{test}}$. We feed $\tilde{\mathbf{x}}_{\text{test}}$ into $p_\theta(\mathbf{x}|\tilde{\mathbf{x}} = \tilde{\mathbf{x}}_{\text{test}})$ and draw samples from the model. We visualize the results in figure 6. As we can see, the model exhibits reasonable denoising results, which shows that the autoregressive model is capable of learning the data distribution when conditioned on the smoothed data.

(a) Noisy MNIST (b) MNIST denoising (c) Noisy CIFAR10 (d) CIFAR10 denoising

Figure 6: Denoising with $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$

Inpainting. Since both $p_\theta(\tilde{\mathbf{x}})$ and $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ are parameterized by an autoregressive model, we can also perform image inpainting using our method. We present the inpainting results on CIFAR-10 in figure 7a and CelebA in figure 7b.

(a) CIFAR10 inpainting

(b) CelebA inpainting

Figure 7: Inpainting results from our two-step method.

Table 2: Inception and FID scores for unconditional CIFAR-10. "Single-step" samples are generated solely by $p_\theta(\tilde{\mathbf{x}})$. "Two-step" samples are generated by drawing samples from $p_\theta(\tilde{\mathbf{x}})$ and then "denoised" by $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$. Although the samples from "single-step" might appear visually similar to samples from the "two-step" method, there is still a gap between their Inception and FID scores.

Model	Inception	FID	BPD
PixelCNN (Oord et al., 2016b)	4.60	65.93	**3.14**
PixelIQN (Ostrovski et al., 2018)	5.29	49.46	-
EBM (Du & Mordatch, 2019)	6.02	40.58	-
i-ResNet (Behrmann et al., 2019)	-	65.01	3.45
MADE (Germain et al., 2015)	-	-	5.67
Glow (Kingma & Dhariwal, 2018)	-	46.90	3.35
Single-step (Ours)	$7.50 \pm .08$	57.53	-
Two-step (Ours)	$\mathbf{7.84} \pm .07$	**29.83**	≤ 3.53

5 Related Work

Our approach is related to two-stage VAE (Dai & Wipf, 2019) which introduces a second VAE to correct the errors made by the first VAE. However, they focus on learning low dimensional manifolds embedded in a high dimensional space and does not apply randomized smoothing to the data distribution. Our approach also shares some similarities with denoising autoencoders (DAE, Vincent et al. (2008)) which recovers a clean data from a corrupted data. However, unlike DAE which has a trainable encoder and a fixed prior distribution, our approach fixes the encoder and models the prior using an autoregressive model. Generative stochastic networks (GSN, Bengio et al. (2014)) use DAEs to train a Markov chain whose equilibrium distribution matches the data distribution. However, GSN needs to start the chain from a sample that is very close to the training distribution.

Denoising diffusion model (Sohl-Dickstein et al., 2015; Ho et al., 2020) and NCSN (Song & Ermon (2019; 2020)) address the issue of GSNs by considering a sequence of distributions corresponding to data corrupted with various noise levels. By setting multiple noise levels that are close to each other, the sample from the previous level can serve as a proper initialization for the sampling process at the next level. In this way, the model can start from a distribution that is easy to model and gradually move to the desired distribution. However, due to the long sequence of noise levels, such approaches require many steps for the chain to converge to the right data distribution.

In this paper, we instead propose to use *only one level* of smoothing by modeling each step with a powerful autoregressive model instead of deterministic autoencoders. Motivated by the success of "randomized smoothing" techniques in adversarial defense (Cohen et al., 2019), we perform randomized smoothing directly to the data distribution. Unlike denoising score matching (Vincent, 2011), a technique closely related to denoising diffusion models and NCSN, which requires the perturbed noise to be a Gaussian distribution, we are able to work with different noise distributions.

6 Discussion

In this paper, we propose to incorporate randomized smoothing techniques into autoregressive modeling. By choosing the smoothness level appropriately, this seemingly simple approach is able to drastically improve the sample quality of existing autoregressive models on several synthetic and real-world datasets while retaining reasonable likelihoods. Our work provides insights into how recent adversarial defense techniques can be leveraged to building more robust generative models. Since we apply randomized smoothing technique directly to the target data distribution other than the model, we believe our approach is also applicable to other generative models such as normalizing flows, variational autoencoders and GANs.

REFERENCES

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582, 2019.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pp. 226–234, 2014.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. *arXiv preprint arXiv:1902.02918*, 2019.

Rob Cornish, Anthony L Caterini, George Deligiannidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. *arXiv preprint arXiv:1909.13833*, 2019.

Bin Dai and David Wipf. Diagnosing and enhancing vae models. *arXiv preprint arXiv:1903.05789*, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, December 2014.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Matti Kääriäinen. Lower bounds for reductions. In *Atomic Learning Workshop*, 2006.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in neural information processing systems*, pp. 4601–4609, 2016.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pp. 10771–10780, 2018.

Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. In *Advances in neural information processing systems*, pp. 1786–1794, 2010.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016a.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016b.

Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive quantile networks for generative modeling. *arXiv preprint arXiv:1806.05575*, 2018.

Martin Raphan and Eero P Simoncelli. Least squares estimation without priors or supervision. *Neural computation*, 23(2):374–420, 2011.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Saeed Saremi and Aapo Hyvarinen. Neural empirical bayes. *Journal of Machine Learning Research*, 20:1–23, 2019.

Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

A PROOFS

Theorem 1. *Given a continuous 1-d distribution $p(x)$ that is supported on $\mathbb{R}$, for any 1-d distribution $q(\tilde{x}|x)$ that is symmetric (*i.e. $q(\tilde{x}|x) = q(x|\tilde{x})$*), stationary (*i.e. *translation invariant) and satisfies $\lim_{x\to\infty} p(x)q(x|\tilde{x}) = 0$ for any given $\tilde{x}$, we have $Lip(q(\tilde{x})) \leq Lip(p_{data}(x))$, where $q(\tilde{x}) \triangleq \int q(\tilde{x}|x)p(x)dx$ and $Lip(\cdot)$ denotes the Lipschitz constant of the given 1-d function.*

Proof. First, we have that:

$$|\nabla_{\boldsymbol{x}} p(\boldsymbol{x})| = |p(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})| \leq Lip(p) \tag{5}$$

and if we assume symmetry, i.e. $q(\boldsymbol{x}|\tilde{\boldsymbol{x}}) = q(\tilde{\boldsymbol{x}}|\boldsymbol{x})$ then by integration by parts we have:

$$\nabla_{\tilde{\boldsymbol{x}}} q(\tilde{\boldsymbol{x}}) = \mathbb{E}_{p(\boldsymbol{x})}[\nabla_{\tilde{\boldsymbol{x}}} q(\tilde{\boldsymbol{x}}|\boldsymbol{x})] = \mathbb{E}_{p(\boldsymbol{x})}[\nabla_{\boldsymbol{x}} q(\boldsymbol{x}|\tilde{\boldsymbol{x}})] = -\mathbb{E}_{p(\boldsymbol{x})}[q(\boldsymbol{x}|\tilde{\boldsymbol{x}})\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})] \tag{6}$$

Therefore,

$$Lip(q) = \max_{\tilde{\boldsymbol{x}}} |-\mathbb{E}_{p(\boldsymbol{x})}[q(\boldsymbol{x}|\tilde{\boldsymbol{x}})\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})]| = \max_{\tilde{\boldsymbol{x}}} |\sum_{\boldsymbol{x}} q(\boldsymbol{x}|\tilde{\boldsymbol{x}})p(\boldsymbol{x})\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x})| \tag{7}$$

$$\leq \max_{\tilde{\boldsymbol{x}}} |\sum_{\boldsymbol{x}} q(\boldsymbol{x}|\tilde{\boldsymbol{x}})Lip(p)| = Lip(p) \max_{\tilde{\boldsymbol{x}}} |\sum_{\boldsymbol{x}} q(\boldsymbol{x}|\tilde{\boldsymbol{x}})| = Lip(p) \tag{8}$$

which proves the result. □

Theorem 2. *Assume that the symmetric and stationary smoothing distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$ has small variance and negligible higher order moments, then*

$$\mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[p_\theta(\tilde{\mathbf{x}})] \approx \mathbb{E}_{p_{data}(\mathbf{x})}\left[\log p_\theta(\mathbf{x}) + \frac{\eta}{2}\sum_i \frac{\partial^2 \log p_\theta}{\partial x_i^2}\right],$$

for some constant η.

Proof. To see this, we first note that the new training objective for the smoothed data distribution is $\mathbb{E}_{p_{\text{data}}}\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log p_\theta(\tilde{\mathbf{x}})]$. Since $q(\tilde{\mathbf{x}}|\mathbf{x})$ is Gaussian, we can write $q(\tilde{\mathbf{x}}|\mathbf{x}) = p(\boldsymbol{\epsilon})$ for some distribution p, where $\boldsymbol{\epsilon} = \tilde{\mathbf{x}} - \mathbf{x}$. Then we have

$$\mathbb{E}_{p_{\text{data}}}\mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})}[\log p_\theta(\tilde{\mathbf{x}})] = \int \log p_\theta(\mathbf{x}+\boldsymbol{\epsilon})p_{\text{data}}(\mathbf{x})p(\boldsymbol{\epsilon})\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\epsilon}, \tag{9}$$

Using Taylor expansion, we have:

$$\log p_\theta(\mathbf{x}+\boldsymbol{\epsilon}) = \log p_\theta(\mathbf{x}) + \sum_i \epsilon_i \frac{\partial \log p_\theta}{\partial x_i} \log p_\theta(\mathbf{x}) + \frac{1}{2}\sum_{i,j} \epsilon_i\epsilon_j \frac{\partial^2 \log p_\theta}{\partial x_i \partial x_j} + o(\epsilon^2).$$

Since that $\boldsymbol{\epsilon}$ is independent of $\mathbf{x}$ and

$$\int \epsilon_i \mathrm{d}\epsilon_i = 0, \quad \int \epsilon_i\epsilon_j \mathrm{d}\epsilon_i \mathrm{d}\epsilon_j = \delta_{i,j}\eta,$$

for some constant η, where $\delta_{i,j}$ is the Kronecker delta function, Equation 9 becomes

$$\mathbb{E}_{p_{\text{data}}}[\log p_\theta(\mathbf{x})] + \int \left(\frac{1}{2}\sum_i \epsilon_i^2 \frac{\partial^2 \log p_\theta}{\partial x_i^2} + o(\epsilon^2)\right) p_{\text{data}}(\mathbf{x})p(\boldsymbol{\epsilon})\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\epsilon} \tag{10}$$

$$= \mathbb{E}_{p_{\text{data}}}\left[\log p_\theta(\mathbf{x}) + \frac{\eta}{2}\sum_i \frac{\partial^2 \log p_\theta}{\partial x_i^2}\right] + \int o(\epsilon^2)p_{\text{data}}(\mathbf{x})p(\boldsymbol{\epsilon})\mathrm{d}\mathbf{x}\mathrm{d}\boldsymbol{\epsilon} \tag{11}$$

where the second term serves as a regularization for the original objective $\mathbb{E}_{p_{\text{data}}}[\log p_\theta(\mathbf{x})]$. □

B IMAGE EXPERIMENTS

B.1 SETTINGS

For the image experiments, we first rescale images to $[-1, 1]$ and then perturb the images with $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}|\mathbf{x}, \sigma^2 I)$. We use $\sigma = 0.5$ for MNIST and $\sigma = 0.3$ for both CIFAR-10 and CelebA. The selection of σ is mainly based on analysis in (Saremi & Hyvarinen, 2019). More specifically, given an image, we consider the median value of the Euclidean distance between two data points in a dataset, and then divide it by $2\sqrt{D}$, where D is the dimension of the data. This provides us with a way of selecting the variance of $q(\tilde{\mathbf{x}}|\mathbf{x})$, when $q(\tilde{\mathbf{x}}|\mathbf{x})$ is a Gaussian distribution. We find this selection of variance able to generate reasonably well samples in practice. We train all the models with Adam optimizer with learning rate 0.0002. To model $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$, we stack $\tilde{\mathbf{x}}$ and $\mathbf{x}$ together at the second dimension to obtain $\hat{\mathbf{x}} = [\tilde{\mathbf{x}}, \mathbf{x}]$, which ensures that $\tilde{\mathbf{x}}$ comes before $\mathbf{x}$ in the pixel ordering. For instance, this stacking would provide an image $\hat{\mathbf{x}}$ with size $1 \times (2 \times 28) \times 28$ on a MNIST image, and an image with size $3 \times (2 \times 32) \times 32$ on a CIFAR-10 image. Since PixelCNN++ consists of convolutional layers, we can directly feed $\hat{\mathbf{x}}$ into the default architecture without modifying the model architecture. As the latter pixels of the input only depend on the previous pixels in an autoregressive model and $\tilde{\mathbf{x}}$ comes before $\mathbf{x}$, we can parameterize $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ by computing the likelihoods only on $\mathbf{x}$ using the outputs from the autoregressive model.

B.2 MORE SAMPLES

Figure 8: CIFAR-10 unconditioned samples from $p_\theta(\tilde{\mathbf{x}})$.

Figure 9: CIFAR-10 unconditioned samples from $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$.

Figure 10: CIFAR-10 unconditioned samples from the original PixelCNN++ method.

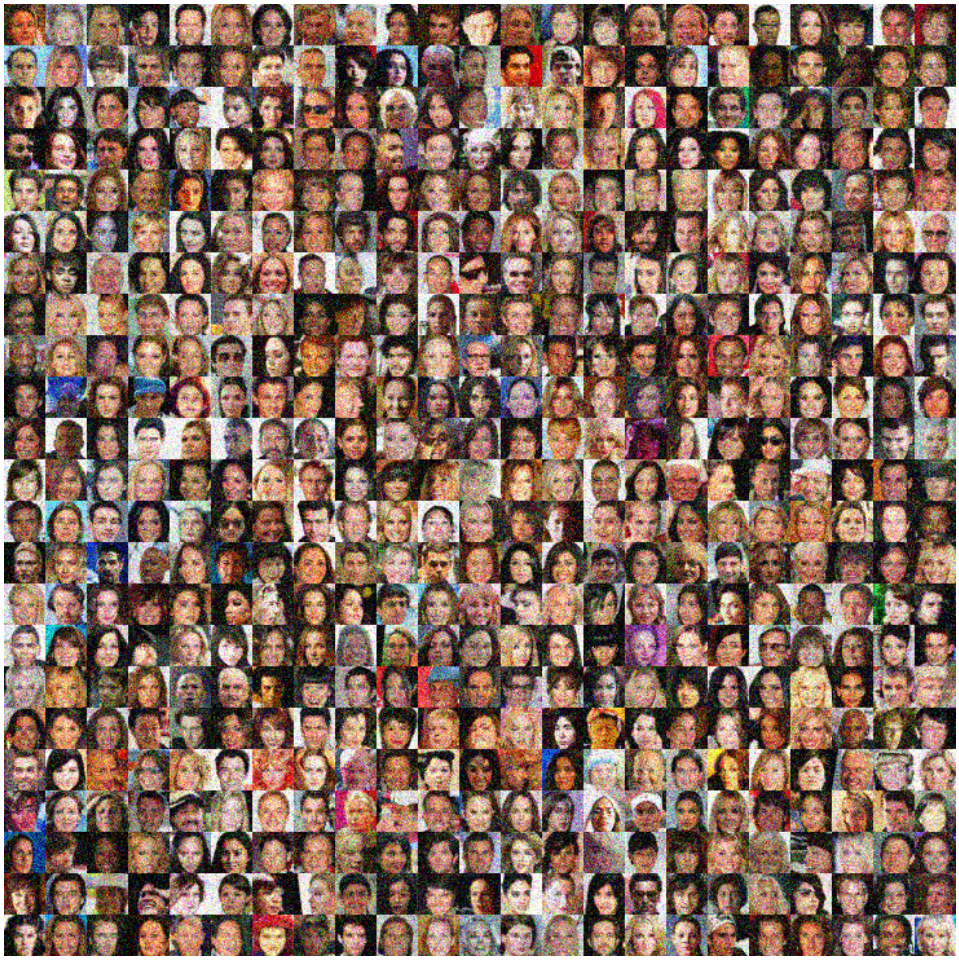


Figure 11: CelebA unconditioned samples from $p_\theta(\tilde{\mathbf{x}})$.

Figure 12: CelebA unconditioned samples from $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$.

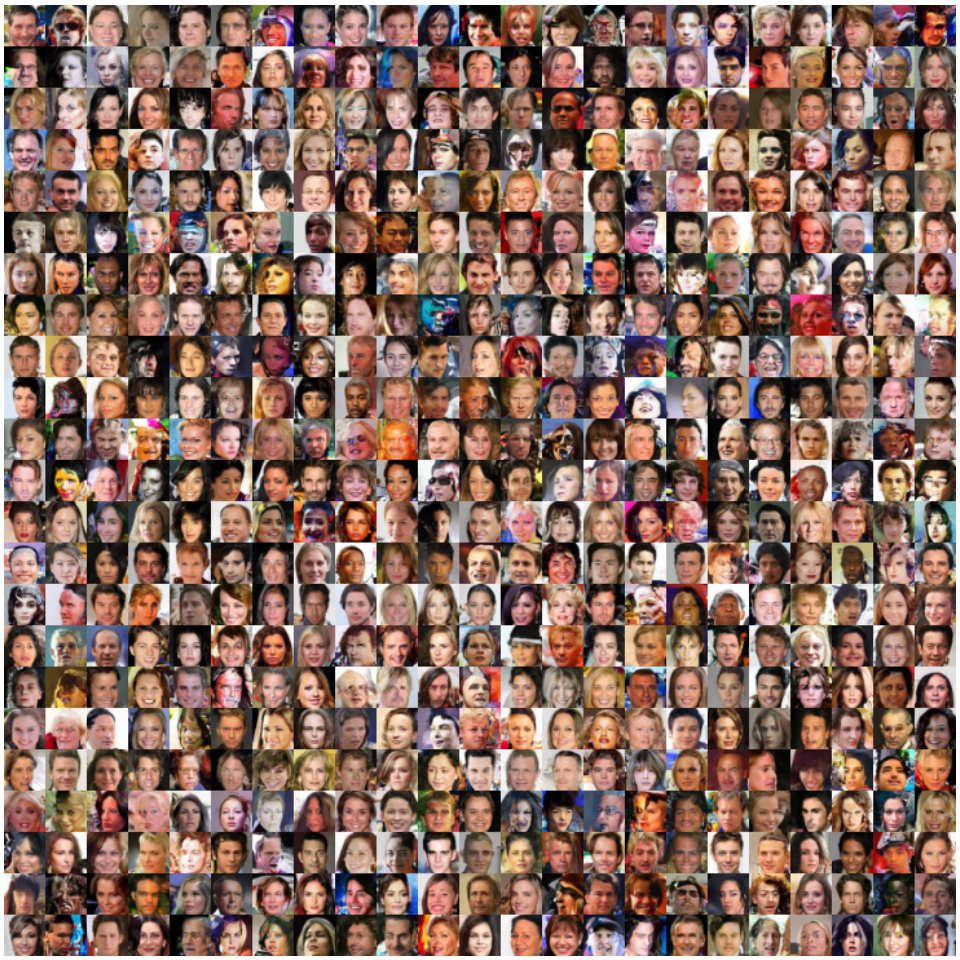


Figure 13: CelebA unconditioned samples from the original PixelCNN++ method.

B.3 NEAREST NEIGHBORS

Figure 14: Nearest neighbors measured by the ℓ_2 distance between images. Images on the left of the red vertical line are samples from our model. Images on the right are nearest neighbors in the training dataset.

Figure 15: Nearest neighbors measured by the ℓ_2 distance in the feature space of an Inception V3 network pretrained on ImageNet. Images on the left of the red vertical line are samples from our model. Images on the right are nearest neighbors in the training dataset.

Figure 16: Nearest neighbors measured by the ℓ_2 distance between images. Images on the left of the red vertical line are samples from our model. Images on the right are nearest neighbors in the training dataset.

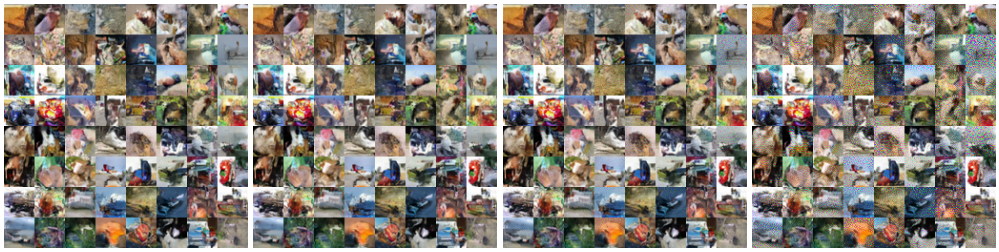
Figure 17: Nearest neighbors measured by the ℓ_2 distance in the feature space of an Inception V3 network pretrained on ImageNet. Images on the left of the red vertical line are samples from our model. Images on the right are nearest neighbors in the training dataset.

B.4 ABLATION STUDIES

In this section, we show that gradient-based "single-step denoising" will not improve sample qualities without performing randomized smoothing. To see this, we draw samples from a PixelCNN++ $p_\theta(\mathbf{x})$ trained directly on $p_{\text{data}}(\mathbf{x})$ (*i.e.* without smoothing). We perform "single-step denoising" update defined as

$$\mathbf{x} = \mathbf{x} + \sigma^2 \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}). \tag{12}$$

We explore various values for σ, and report the results in figure 18. This shows that "single-step denoising" alone (without randomized smoothing) will not improve sample quality of PixelCNN++.



(a) $\sigma = 0$ (Original) (b) $\sigma = 0.01$ (c) $\sigma = 0.05$ (d) $\sigma = 0.1$

Figure 18: "Single-step denoising" on PixelCNN++ trained on un-smoothed data. $\sigma = 0$ corresponds to the original samples.