

# Improved Autoregressive Modeling with Distribution Smoothing

Mikhail Kuznetsov

CMC MSU

February 12, 2021

Let  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  —  $D$ -dimensional i.i.d samples from a continuous data distribution  $p_{\text{data}}(\mathbf{x})$

## Property

An autoregressive model decomposes a joint distribution into univariate conditionals [2]:

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

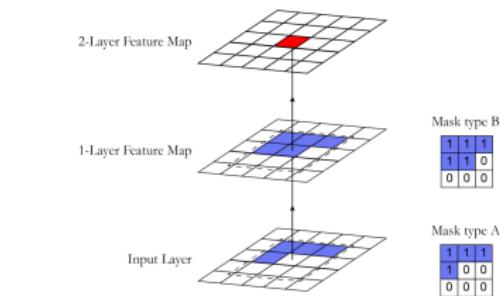
## Goal

Find  $\theta$  — parameters of the model such that  $p_\theta(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$

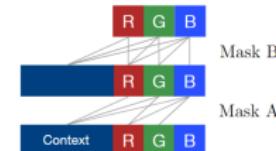
A commonly used approach for density estimation is maximum likelihood estimation (MLE), i.e., by maximizing

$$L(\theta) \triangleq \frac{1}{N} \sum_{i=1}^N \log p_\theta(\mathbf{x}_i)$$

# PixelCNN [3]

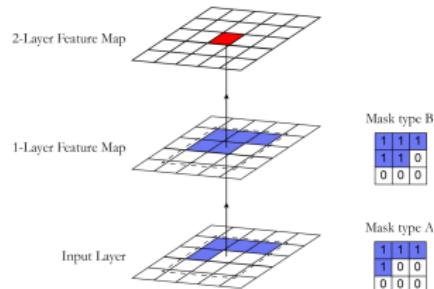
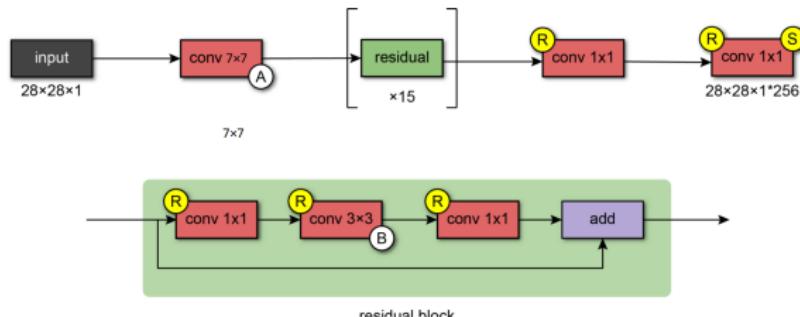


Masking 1



Masking 2

# PixelCNN [3]



## Legend

### Layers

conv 3x3 Convolutional operations, in red

add Merge operations, in purple

A Masked layer, type A

B Masked layer, type B

### Functions

On the right side = pos layer operation; on the left side = pre layer operation

R ReLU

S Softmax

### Blocks

Blocks in green. The operations that make up these blocks will also be shown.

$$\text{block A} = \text{conv 3x3} \rightarrow \text{conv 1x1}$$

### Repeated layers or blocks

$$\left[ \text{block A} \right]_{\times 2} = \text{block A} \rightarrow \text{block A}$$

## Architecture of the PixelCNN

Image source

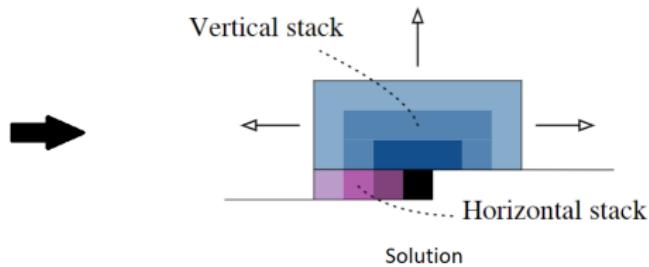
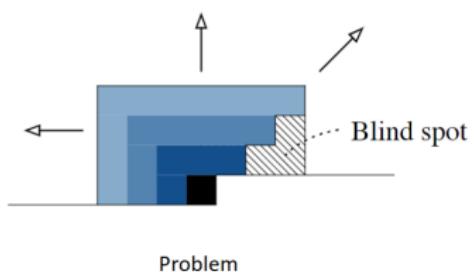
## Masking 1



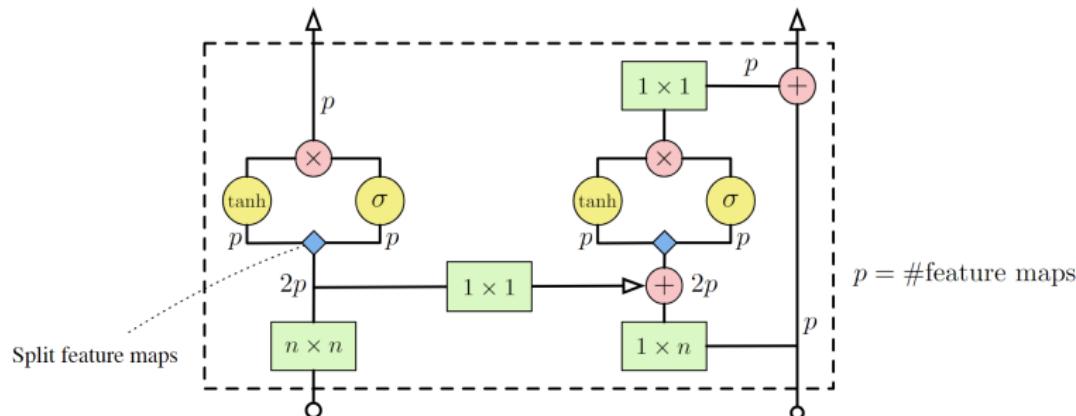
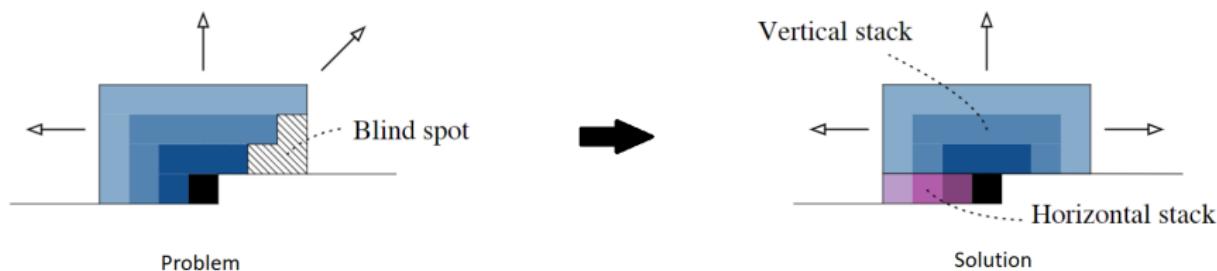
## Masking 2



# Gated PixelCNN [4]



# Gated PixelCNN [4]



A single layer in the Gated PixelCNN architecture

# PixelCNN++ [5]

The most important modifications:

# PixelCNN++ [5]

The most important modifications:

- discretized logistic mixture likelihood

# PixelCNN++ [5]

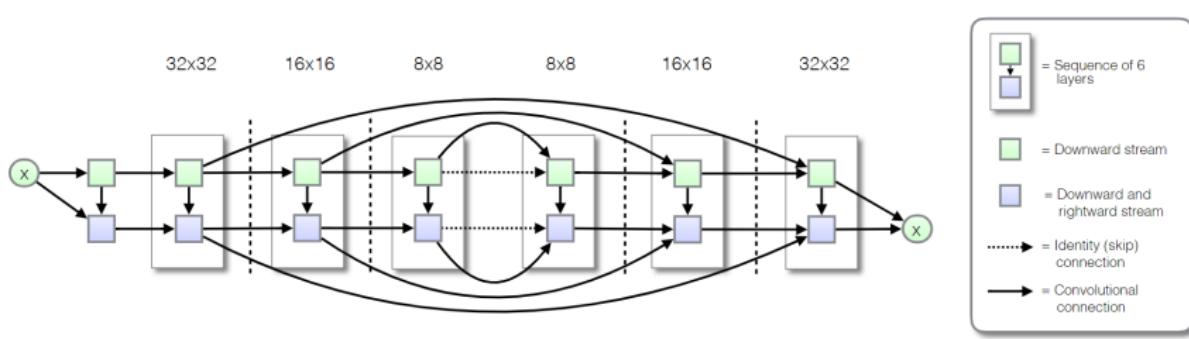
The most important modifications:

- discretized logistic mixture likelihood
- conditioning on whole pixels

# PixelCNN++ [5]

The most important modifications:

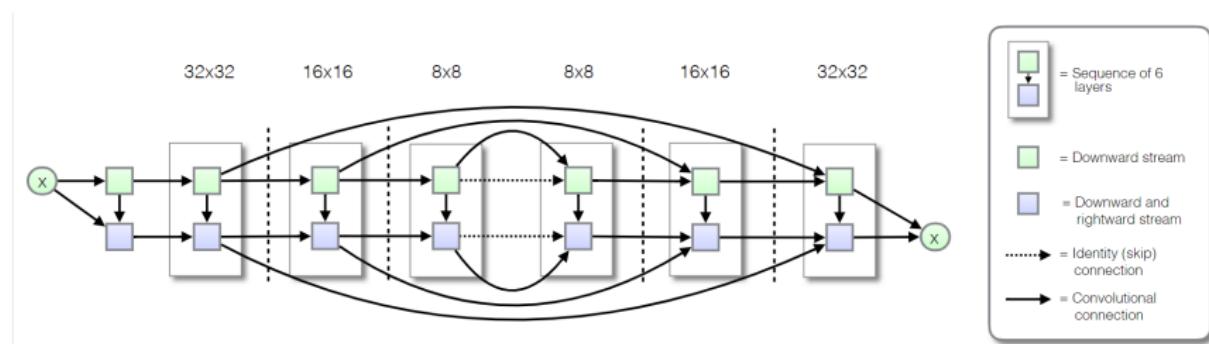
- discretized logistic mixture likelihood
- conditioning on whole pixels
- dilated convolution → convolution with stride



# PixelCNN++ [5]

The most important modifications:

- discretized logistic mixture likelihood
- conditioning on whole pixels
- dilated convolution → convolution with stride

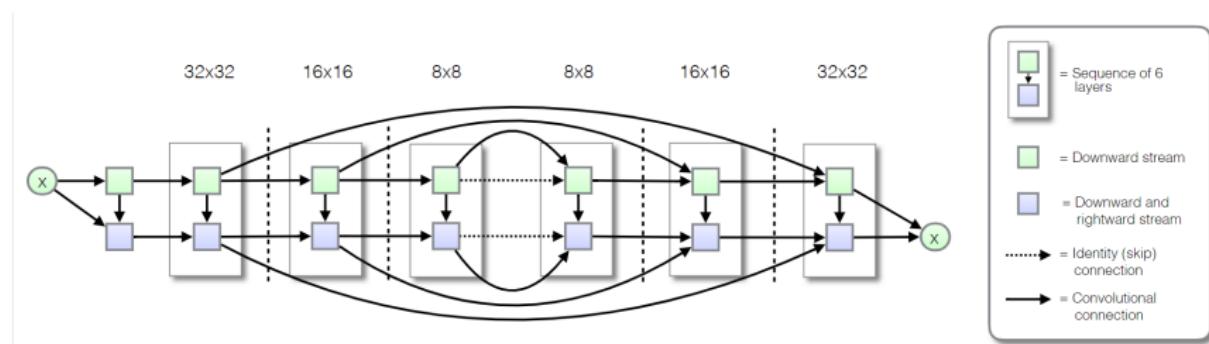


- adding short-cut connections

# PixelCNN++ [5]

The most important modifications:

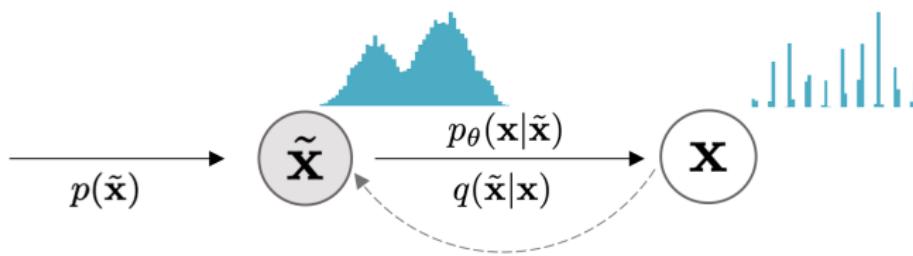
- discretized logistic mixture likelihood
- conditioning on whole pixels
- dilated convolution → convolution with stride



- adding short-cut connections
- regularization using dropout

For complete details see [here](#)

# General idea



Overview of the method

( $\tilde{\mathbf{x}}$  — the smoothed data,  $q(\tilde{\mathbf{x}} | \mathbf{x})$  — the smoothing distribution) [1]

# Problem 1: Manifold hypothesis

Many real world data distributions (e.g. natural images) **may lie in the vicinity of a low-dimensional manifold** and can often have complicated densities with sharp transitions (i.e. high Lipschitz constants) that are difficult to model.

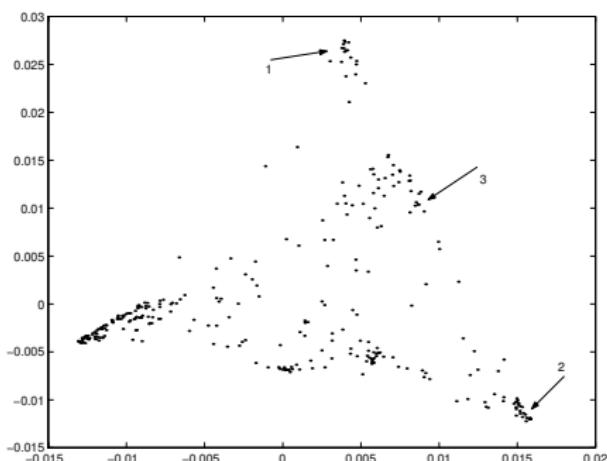


Figure (a): The 300 most frequent words of the Brown corpus represented in the spectral domain.

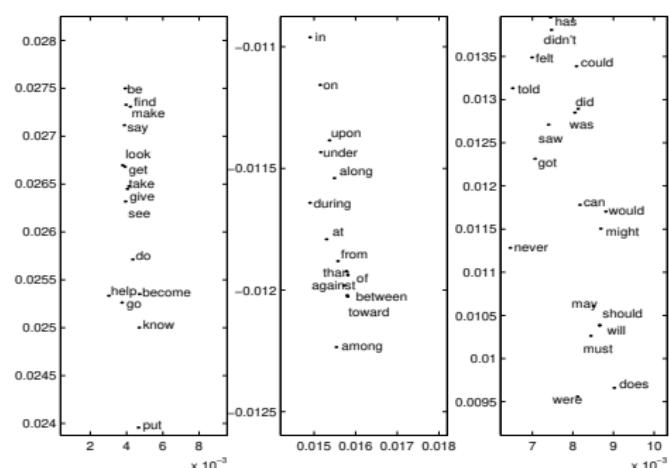
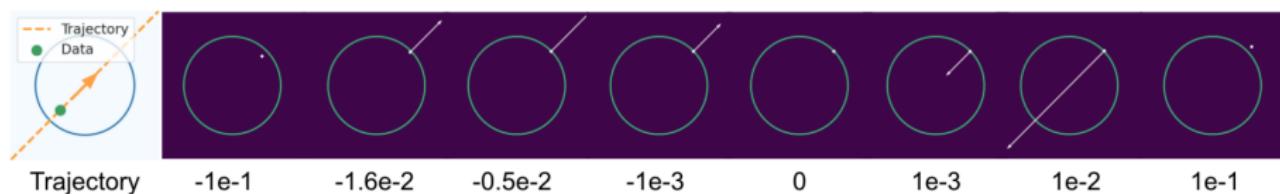


Figure (b): Fragments labeled by arrows: (left) infinitives of verbs, (middle) prepositions, and (right) mostly modal and auxiliary verbs.

Source: (Belkin M., Niyogi P., 2003) [6]

# Problem 1: Manifold hypothesis



Ring distribution (almost a unit circle) formed by rotating the 1-d Gaussian distribution  $\mathcal{N}(1, 0.01^2)$  around the origin. The data location for each figure is  $(\sqrt{0.5} + c, \sqrt{0.5} + c)$ , where  $c$  is the number below each figure and  $(\sqrt{0.5}, \sqrt{0.5})$  is the upper right intersection of the trajectory with the unit circle. [1]

## Problem 2: Compounding errors

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

Compounding errors comes from the inaccurate approximation of the conditional distributions.

The reasons for this may be:

- Curse of dimensionality + very limited amount of training data (in some cases)
- The current state is based on the values of the previous states
- Adversarial attacks

## Theorem 1 Let:

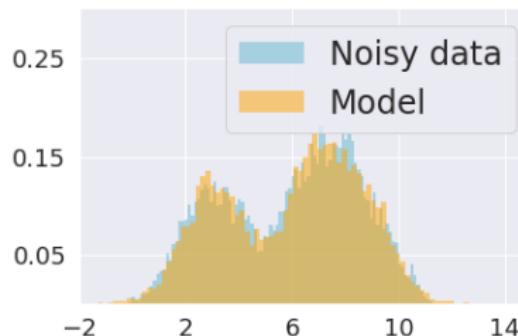
- 1  $p(x)$  — a continuous 1-d distribution that is supported on  $\mathbb{R}$
- 2  $q(\tilde{x} | x)$  — 1-d distribution that is:
  - symmetric (i.e.  $q(\tilde{x} | x) = q(x | \tilde{x})$ )
  - stationary (i.e. translation invariant)
  - $\lim_{x \rightarrow \infty} p(x)q(x | \tilde{x}) = 0$  for any given  $\tilde{x}$

Then:  $\text{Lip}(q(\tilde{x})) \leq \text{Lip}(p_{\text{data}}(x))$ ,

where  $q(\tilde{x}) \triangleq \int q(\tilde{x} | x)p(x)dx$  and  $\text{Lip}(\cdot)$  denotes the Lipschitz constant of the given  $1 - d$  function.



(a) Data



(b) Smoothed data

## Proposition 1 (Informal) Let:

1  $q(\tilde{x} | x)$  is such that:

- symmetric
- stationary
- has small variance
- has negligible higher order moments (i.e. very small)

Then:

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

for some constant  $\eta$ .

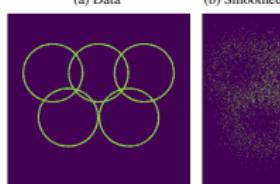
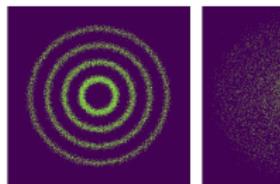
Since the samples from  $p_{\text{data}}$  should be close to a local maximum of the model, this encourages the second order gradients computed at a data point  $x$  to become closer to zero (if it were positive then  $x$  will not be a local maximum), creating a smoothing effect.

# Introducing 2<sup>nd</sup> distribution: $p_\theta(x | \tilde{x})$

**Case 1**  $q(\tilde{x} | x) = \mathcal{N}(\tilde{x} | x, \sigma^2 I)$  and  $\sigma$  is small

## Single-step denoising [7]

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



Example of "single-step denoising"

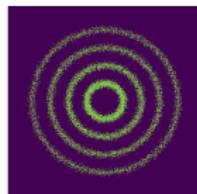


Figure 26: "Single-step denoising" on PixelCNN++ trained on unsmoothed data.  $\sigma = 0$  corresponds to the original samples

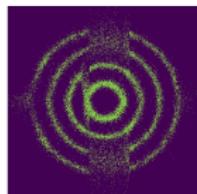
## Case 2 General case

### Maximizing an evidence lower bound (ELBO)<sup>1</sup>

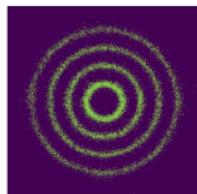
$$\log p_\theta(x) \geq \mathbb{E}_{p_{\text{data}}(x)} [\mathbb{E}_{q(\tilde{x}|x)} [\log p_\theta(\tilde{x})] - \mathbb{E}_{q(\tilde{x}|x)} [\log q(\tilde{x} | x)] + \mathbb{E}_{q(\tilde{x}|x)} [\log p_\theta(x | \tilde{x})]]$$



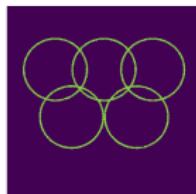
(a) Rings



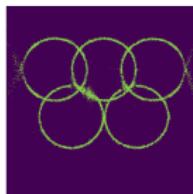
(b) MADE (6)



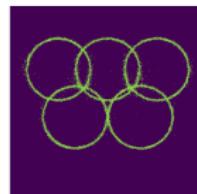
(c) Ours (3)



(d) Olympics



(e) MADE (6)



(f) Ours (3)

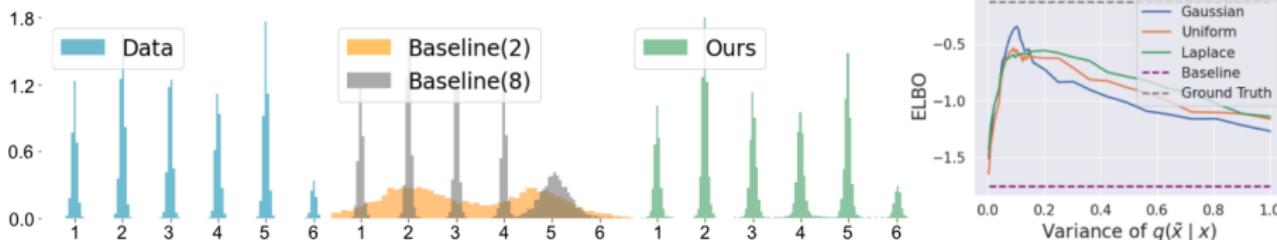
We use a MADE model with comparable number of parameters for both our method and the baseline. (the number mixture of logistics)

Dataset	RealNVP	CIF-RealNVP	MADE (3 mixtures)	MADE (6 mixtures)	Ours (3 mixtures)
Rings	2.81	2.81	3.26	2.81	<b>2.71</b>
Olympics	1.80	1.74	1.27	<b>0.80</b>	<b>0.80</b>

Negative log-likelihoods on 2-d synthetic datasets (lower is better))

<sup>1</sup>Formula derivation

# Choosing the smoothing distribution



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.

For each type of distribution, we perform a grid search to find the optimal variance. Since our approach requires the modeling of both  $p_\theta(\tilde{x})$  and  $p_\theta(x | \tilde{x})$ , we stack  $\tilde{x}$  and  $x$  together, and use a MADE model [8] with a mixture of two logistic components to parameterize  $p_\theta(\tilde{x})$  and  $p_\theta(x | \tilde{x})$  at the same time. For the baseline model, we train a mixture of logistics model directly on  $p_{\text{data}}(x)$ .



Column 1

Column 2

Column 3

Column 4

From left to right: **Column 1**: samples from  $p_{\theta}(\tilde{x})$ . **Column 2**: "single-step denoising" samples from  $p_{\theta}(\tilde{x})$ . **Column 3**: samples from  $p_{\theta}(x | \tilde{x})$ . **Column 4**: samples from the baseline PixelCNN++ model with parameters comparable to the sum of total parameters of  $p_{\theta}(\tilde{x})$  and  $p_{\theta}(x | \tilde{x})$ .

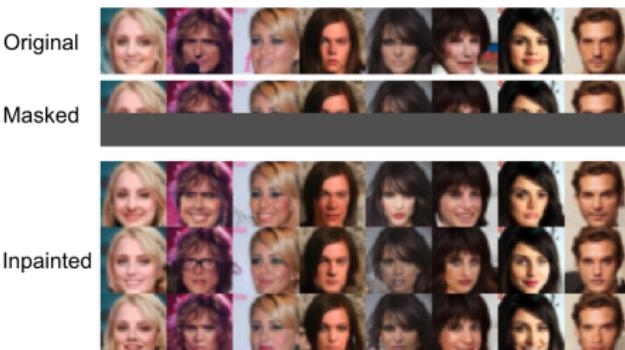
Model	Inception $\uparrow$	FID $\downarrow$	BPD $\downarrow$
PixelCNN (Oord et al., 2016b)	4.60	65.93	<b>3.14</b>
PixelIQN (Ostrovski et al., 2018)	5.29	49.46	-
EBM (Du & Mordatch, 2019)	6.02	40.58	-
i-ResNet (Behrmann et al., 2019)	-	65.01	3.45
MADE (Germain et al., 2015)	-	-	5.67
Glow (Kingma & Dhariwal,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

Trained on unconditional CIFAR-10. Inception (higher is better) and FID scores (lower is better). "Single-step" samples are generated solely by  $p_\theta(\tilde{x})$ . "Two-step" samples are generated by drawing samples from  $p_\theta(\tilde{x})$  and then "denoised" by  $p_\theta(x | \tilde{x})$ .

We use PixelCNN++ (Salimans et al., 2017) as the model architecture for both  $p_\theta(\tilde{x})$  and  $p_\theta(x | \tilde{x})$ . We provide more details about settings in Appendix C.

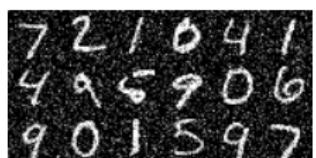


(a) CIFAR10 inpainting

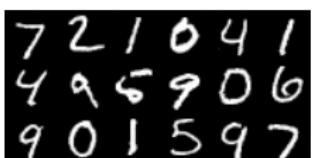


(b) CelebA inpainting

Inpainting results from our two-step method



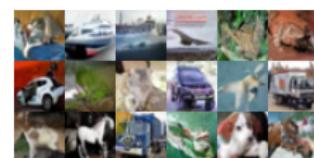
(a) Noisy MNIST



(b) MNIST denoising



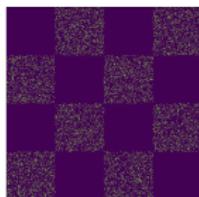
(c) Noisy CIFAR10



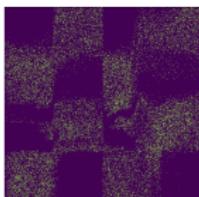
(d) CIFAR10 denoising

Denoising with  $p_\theta(x | \tilde{x})$

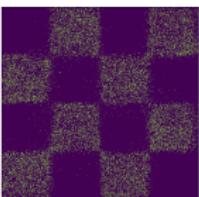
# Normalizing flows [9]



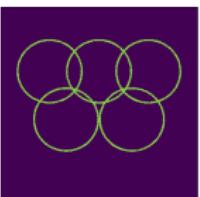
(a) Data



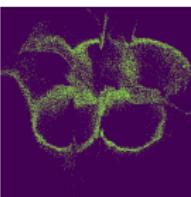
(b) RealNVP



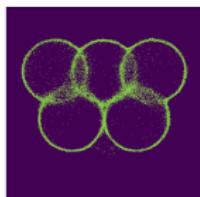
(c) Ours



(d) Data



(e) RealNVP



(f) Ours

We compare the RealNVP model trained with randomize smoothing, where we use  $p_\theta(x | \tilde{x})$  (also a RealNVP) to revert the smoothing process, with a RealNVP trained with the original method but with comparable number of parameters.

Model/Dataset	checkerboard	Olympics
Original	3.72	1.32
Ours	<b>3.64</b>	<b>1.80</b>

NLL on the datasets. Distribution are modeled as RealNVP

- [1] Improved autoregressive modeling with distribution smoothing.  
<https://openreview.net/pdf?id=rJA5Pz71HKb>, 2021.
- [2] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma.  
Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications.  
[arXiv:1701.05517](https://arxiv.org/pdf/1701.05517.pdf), 2017.
- [3] A aron van den Oord, Nal Kalchbrennern, and Koray Kavukcuoglu.  
Pixel recurrent neural networks.  
<https://arxiv.org/pdf/1601.06759.pdf>, 2016.
- [4] A aron van den Oord, Nal Kalchbrennern, Koray Kavukcuoglu et al.  
Conditional image generation with pixelcnn decoders.  
<https://arxiv.org/pdf/1606.05328.pdf>, 2016.
- [5] Tim Salimansn, Andrej Karpathyn, Xi Chenn, Diederik P. Kingmas.  
Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications.  
<https://arxiv.org/pdf/1701.05517.pdf>, 2017.

[6] Mikhail Belkin, Partha Niyogi.

Laplacian eigenmaps for dimensionality reduction and data representation.

[https:](https://www2.imm.dtu.dk/projects/manifold/Papers/Laplacian.pdf)

//www2.imm.dtu.dk/projects/manifold/Papers/Laplacian.pdf, 2003.

[7] Saeed Saremi, Arash Mehrjou, Bernhard Scholkopf, and Aapo Hyvärinen.

Deep energy estimator networks.

arXiv:1805.08306, 2018.

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

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