Locally Masked Convolution for Autoregressive Models

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@berkeley.edu

Deepak Pathak
Carnegie Mellon University
dpathak@cs.cmu.edu

Abstract

High-dimensional generative models have many applications including image compression, multimedia generation, anomaly detection and data completion. State-of-the-art estimators for natural images are autoregressive, decomposing the joint distribution over pixels into a product of conditionals parameterized by a deep neural network, *e.g.* a convolutional neural network such as the PixelCNN. However, PixelCNNs only model a single decomposition of the joint, and only a single generation order is efficient. For tasks such as image completion, these models are unable to use much of the observed context. To generate data in arbitrary orders, we introduce LMConv: a simple modification to the standard 2D convolution that allows arbitrary masks to be applied to the weights at each location in the image. Using LMConv, we learn an ensemble of distribution estimators that share parameters but differ in generation order, achieving improved performance on whole-image density estimation (2.89 bpd on unconditional CIFAR10), as well as globally coherent image completions. Our code is available at https://ajayjain.github.io/lmconv.

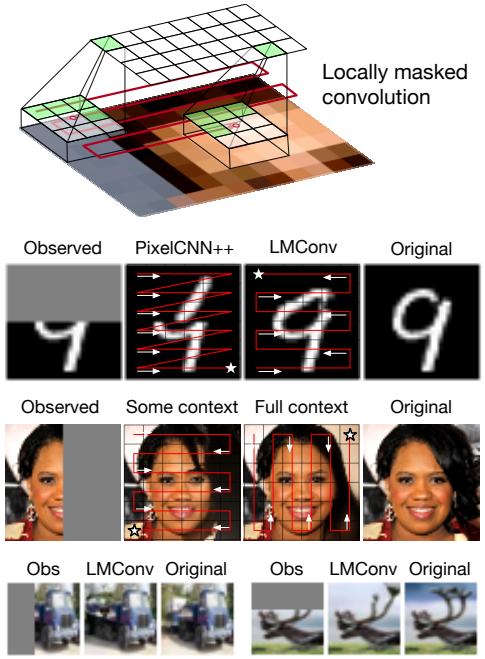


Figure 1: The ideal autoregressive joint distribution decomposition and sampling order are task-dependent. We learn to generate images under multiple orderings with the same parameters via *locally masked convolutions* (top), enabling global coherence for image completion (bottom).

1 INTRODUCTION

Learning generative models of high-dimensional data such as images is a holy grail of machine learning with pervasive applications. Significant progress on this problem would naturally lead to a wide range of applications, including multimedia generation, compression, probabilistic time series forecasting, representation learning, and missing data completion. Many generative modeling frameworks have been proposed. Current state-of-the-art models for high-dimensional image data include (a) autoregressive models (Bengio and Bengio, 2000; Efros and Leung, 1999), (b) normalizing flow density estimators (Rezende and Mohamed, 2015), (c) generative adversarial networks (GANs) (Goodfellow et al., 2014), (d) latent variable models such as the VAE (Kingma and Welling, 2014; Rezende et al., 2014) and (e) energy-based models *e.g.* Hinton (2002); LeCun et al. (2006); Du and Mordatch (2019); Song and Ermon (2019). While GANs, VAEs and EBMs have had great success in high-dimensional image generation, exact likelihoods are generally intractable. Likelihood estimation is key for many

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

practical applications from uncertainty estimation, robustness, reliability and safety perspectives. In contrast, autoregressive and flow models estimate exact likelihoods and can be used for uncertainty estimation, though still have room for improved generation quality. In this work, our focus is on autoregressive models.

Given n variables, one can generate $n!$ autoregressive decompositions of the joint likelihood, each corresponding to a forward sampling order, and more if we assume conditional independence. Early autoregressive texture synthesis (Popat and Picard, 1993; Efros and Leung, 1999) work could support multiple orders. However, recent CNN-based autoregressive models for images (van den Oord et al., 2016b,a; Salimans et al., 2017) capture only one of these orders (typically left-to-right raster scan, Fig. 2) for practical computational efficiency. Training and testing with a single order will not support all scenarios. Consider the image completion task in first row of Figure 1. If the top half of the image is missing, a raster scan generation order from left-to-right and top-to-bottom does not allow the model to condition on the context given in the observed bottom half of the image as the required conditionals are not estimated by the model.

In this work, we propose a scalable, yet simple modification to convolutional autoregressive models to estimate more accurate likelihoods with a minor change in computation during training. Our goal is to support arbitrary orders in a scalable manner, allowing more precise likelihoods by averaging over several graphical models corresponding to orders (a form of Bayesian model averaging). Some past works have supported arbitrary orders in autoregressive models by learning separate parameters for each model (Frey, 1998), or by masking the input image to hide successor variables (Larochelle and Murray, 2011). A more efficient approach is to estimate densities in parallel across dimensions by masking network weights (Germain et al., 2015) differently for each order. However, all these methods are still computationally inefficient and difficult to scale beyond fully-connected networks to convolutional architectures.

In this work, we perform order-agnostic distribution estimation for natural images with state-of-the-art convolutional architectures. We propose to support arbitrary orderings by introducing masking at the level of features, rather than on inputs or weights. We show how an autoregressive CNN can support and learn multiple orders, with a single set of weights, via *locally masked convolutions* that efficiently apply location-specific masks to patches of each feature map. These local convolutions can be efficiently implemented purely via matrix multiplication by incorporating masking at the level of the im2col and col2im separation of convolution (Jia et al., 2014).

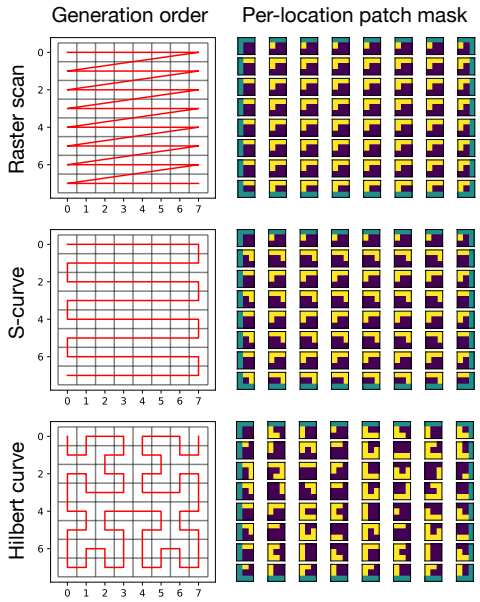


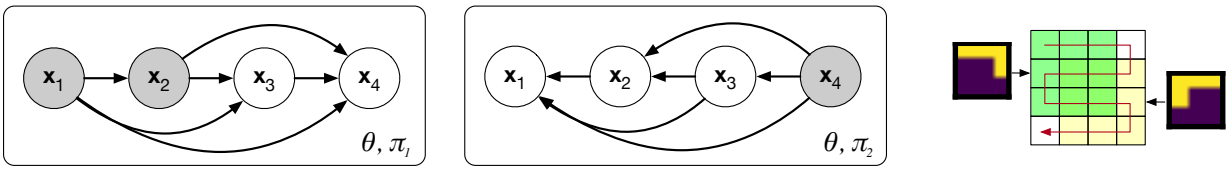
Figure 2: The three pixel generation orders and corresponding local masks that we consider in this work.

Arbitrary orders allow us to customize the traversal based on the needs of the task, which we evaluate in experiments. For instance, consider the examples shown in Fig. 1. The flexibility allows us to select the sampling order that exposes the maximum possible context for image completion, choose orderings that eliminate blind-spots (unobservable pixels) in image generation, and ensemble across multiple orderings using the same network weights. Note that such a model is able to support these image completions without training on any inpainting masks.

In experiments, we show that our approach can be efficiently implemented and is flexible without sacrificing the overall distribution estimation performance. By introducing order-agnostic training via LMConv, we significantly outperform PixelCNN++ on the unconditional CIFAR10 dataset, achieving code lengths of 2.89 bits per dimension. We show that the model can generalize to some novel orders. Finally, we significantly outperform raster-scan baselines on conditional likelihoods relevant to image completion by customizing the generation order.

2 BACKGROUND

Deep autoregressive models estimate high-dimensional data distributions using samples from the joint distribution over D-dimensions $p_{\text{data}}(\mathbf{x}_1, \ldots, \mathbf{x}_D)$. In this setting, we wish to approximate the joint with a parametric model $p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_D)$ by minimizing KL-divergence $D_{KL}(p_{\text{data}}||p_\theta)$, or equivalently by maximizing the log-likelihood of the samples. As a general modeling princi-



(a) Graphical model for suffix completion (b) Graphical model for prefix completion (c) Locally masked convolutions

Figure 3: (a) A graphical model where the final, unobserved variables x_3, x_4 can be efficiently completed via forward sampling conditioned on the observed variables x_1, x_2. (b) When x_4 is observed, we sample x_1, x_2, and x_3 in the second graphical model using the same parameters. (c) LMConv defines the model with masks at each filter location.

ple, we can divide high-dimensional variables into many low-dimensional parts such as single dimensions, and capture dependencies between dimensions with a directed graphical model. Following the notation of (Kingma et al., 2019), these autoregressive (AR) models represent the joint distribution as a product of conditionals,

$$\begin{aligned} p_\theta(\mathbf{x}) &= p_\theta(x_1, \ldots, x_D) \\ &= p_\theta(x_{\pi(1)}) \prod_{i=2}^{D} p_\theta\left(x_{\pi(i)} \mid Pa(\mathbf{x}_{\pi(i)})\right) \end{aligned} \tag{1}$$

where $\pi : [D] \to [D]$ is a permutation defining an order over the dimensions, $Pa(\mathbf{x}_{\pi(i)}) = \mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(i-1)}$ defines the parents of $x_{\pi(i)}$ in the graphical model, and θ is a parameter vector. As any joint can be decomposed in this manner according to the product rule, this factorization provides the foundation for many models including ours. The primary challenge in autoregressive models is defining a sufficiently expressive family for the conditionals where parameter estimation is efficient. Deep autoregressive models parameterize the conditionals with a neural network that is provided the context $Pa(\mathbf{x}_{\pi(i)})$.

Decomposition (1) converts the joint modeling problem into a sequence modeling problem. Forward (ancestral) sampling draws root variable $x_{\pi(1)}$ first, then samples the remaining dimensions in order $x_{\pi(2)}, \ldots, x_{\pi(D)}$ from their respective conditionals. Given a particular autoregressive decomposition of the joint, forward sampling supports a single data generation order. The joint model density for an observed variable can be computed exactly by evaluating each conditional, allowing density estimation and maximum likelihood parameter estimation,

$$\begin{aligned} \mathcal{L}(\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \sum_{i=1}^{D} \log p_\theta\left(x_{\pi(i)} \mid x_{\pi(1)}, \ldots, x_{\pi(i-1)}\right) \\ \theta^* &= \arg_\theta \max \mathcal{L}(\theta) \end{aligned} \tag{2}$$

With some choices of network architecture, the conditionals can be computed in parallel by masking weights (Germain et al., 2015; van den Oord et al., 2016b). In the PixelCNN model family, masked convolutions are *causal*: the features output by a masked convolution can only depend on features earlier in the order.

While the choice of order is arbitrary, temporal and sequential data modalities have a natural ordering from the first dimension in the sequence to the last. For spatial data such as images, a natural ordering is not clear. For computational reasons, a *raster scan* order is generally used where the top left pixel is modeled unconditionally and generation proceeds in row-major fashion across each row from left to right, depicted in Figure 1, second column.

3 IMAGE COMPLETION WITH MAXIMUM RECEPTIVE FIELD

For estimating the distribution of 2D images, a raster scan ordering is perhaps as good of an order as any other choice. That said, the raster scan order has necessitated architectural innovations to allow the neural network to access information far back in the sequence such as two-dimensional PixelRNNs (van den Oord et al., 2016b), two-stream shift-based convolutional architectures (van den Oord et al., 2016a), and self-attention combined with convolution (Chen et al., 2018). These structures significantly improve test-set likelihoods and sample quality, but marry network architectures to the raster scan order.

Fixing a particular order is limiting for missing data completion tasks. Letting $\pi(i) = i$ denote the raster scan order, PixelRNN and PixelCNN architectures can complete only the bottom part of the image via forward sampling: given observations $x_1, \ldots, x_d$, raster scan autoregressive models sequentially sample,

$$\hat{x}_i \sim p_\theta(x_i \mid x_1, \ldots, x_d, \hat{x}_{d+1}, \ldots, \hat{x}_{i-1}). \tag{3}$$

If all dimensions other than x_i are observed, ideally we would sample $\hat{x}_i$ using maximum conditioning context,

$$\hat{x}_i \sim p_\theta(x_i \mid x_{<i}, x_{>i}). \tag{4}$$

Unfortunately, the raster scan model only predicts distributions of the form $p_\theta(x_i \mid x_{<i})$, and ignores observations $x_{>i}$ during completion. In the worst case, a model with

a raster scan generation order *cannot observe any of the context* for an inpainting task where the top half of the image is unknown (Figure 1, PixelCNN++). This leads to image completions that do not respect global structure. Small numbers of dimensions could be sampled by computing the posterior, *e.g.* for $i = 1$,

$$p_\theta(\hat{x}_1 \mid x_{>1}) = \frac{p_\theta(\hat{x}_1, x_{>1})}{\sum_{x'_1} p_\theta(x'_1, x_{>1})}, \tag{5}$$

but this is expensive as each summand requires neural network evaluation, and becomes intractable when several dimensions are unknown. Instead of approximating the posterior, we estimate parameters θ that achieve high likelihood with multiple autoregressive decompositions,

$$\begin{aligned} \mathcal{L}_{\text{OA}}(\theta) &= \mathbb{E}_{x \sim p_{\text{data}}} \mathbb{E}_{\pi \sim p_\pi} \log p_\theta\left(x_1, \ldots, x_D; \pi\right) \\ \theta^* &= \arg_\theta \max \mathcal{L}_{\text{OA}}(\theta) \end{aligned} \tag{6}$$

with p_π denoting a uniform distribution over several orderings. The joint distribution under π factorizes according to (1). The resulting conditionals are all parameterized by the same neural network. By choosing order prior p_π that supports a π such that $\pi(D) = i$, we can use the network with such an ordering to query (4) directly.

During optimization with stochastic gradient descent, we make single-sample estimates of the inner expectation in (6) according to order-agnostic training (Uria et al., 2014; Germain et al., 2015), using a single order per batch.

For a test-time task where $\{x_i : i \in T_{\text{obs}}\}$ are observed, we select a π that the model was trained with such that

$$\{\pi(1), \ldots, \pi(|T_{\text{obs}}|)\} = T_{\text{obs}},$$

i.e. the first $|T_{\text{obs}}|$ dimensions in the generation order are the observed dimensions, then sample according to the rest of the order so that the model posterior over each unknown dimension is conditioned either on observed or previously sampled dimensions.

4 LOCAL MASKING

In this section, we develop *locally masked convolutions* (LMConv): a modification to the standard convolution operator that allows control over generation order and parallel computation of conditionals for evaluating likelihood. In the first convolutional layer of a neural network, C_{out} filters of size $k \times k$ are applied to the input image with spatial invariance: the same parameters are used at all locations in a sliding window. Each filter has $k^2 * C_{\text{in}}$ parameters. For images with discretized intensities, convolutional autoregressive networks transform a spatial $H \times W$, multi-channel image into a tensor of log-probabilities that define the conditional distributions

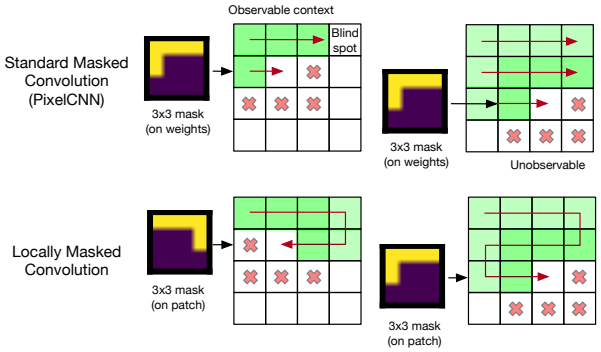


Figure 4: A comparison of standard weight masked convolutions and the proposed locally masked convolution.

of (1). These log-probabilities take the form of an $H \times W$ image, with channel count equal to the number of color channels times the number of bins per color channel. The output log-probabilities at coordinate i, j in the output define the distribution $p_\theta(x_{i,j} \mid Pa(p(x_{i,j}))$. Critically, this distribution must not depend on observations of successors in the Bayesian network, or the product of conditionals will not define a valid distribution due to cyclicity.

NADE (Larochelle and Murray, 2011) circumvents the problem by masking the input image, though requires independent forward passes to compute each factor of the autoregressive decomposition (1). Instead, the PixelCNN model family controls information flow through the network by setting certain weights of the convolution filters to zero, similar to how MADE (Germain et al., 2015) masks the weight matrices in fully-connected layers. We depict masked convolutions for the first convolutional layer in Figure 4. As a single mask is applied to the $C_{\text{in}} \times k \times k$ parameter tensor defining each convolutional filter, the same masking pattern is applied at all locations in the image. Sharing the masking pattern constrains the possible orders, and leads to blind spots which the output distribution is unable to observe.

In practice, convolutions are implemented through general matrix multiplication (GEMM) due to widely available, heavily optimized and parallelized implementations of the operation on GPU and CPU. To use matrix multiplication, the input to a layer is rearranged in memory via the im2col algorithm, which extracts $C_{\text{in}} \times k \times k$ patches from the $C_{\text{in}} \times H \times W$ input at each location that a convolutional filter will be applied. Assuming padding and a stride of 1 is used, the rearrangement yields matrix X with $C_{\text{in}} * k^2$ rows and $H * W$ columns. To perform convolution, the framework left-multiplies weight matrix $\mathcal{W}$, storing $Y = \mathcal{W}X$, adds a bias, and finally rearranges Y into a spatial format via the col2im algorithm.

We exploit this data rearrangement to arbitrarily mask the input to the convolutional filter at each location it is

Algorithm 1 LMConv: Locally masked 2D convolution

Input: image x, weights $\mathcal{W}$, bias b, generation order π. x is $B \times C_{\text{in}} \times H \times W$ dimensional and $\mathcal{W}$ is $C_{\text{out}} \times C_{\text{in}} * k_1 * k_2$ dimensional
Create mask matrix $\mathcal{M}$ with Algorithm 2
Extract patches: $X = \texttt{im2col}(\texttt{pad}(x), k_1, k_2)$
Mask patches: $X = \mathcal{M} \odot X$
Perform convolution via batch MM: $Y = \mathcal{W}X + b$
Assemble patches: $y = \texttt{col2im}(Y)$
return y

Algorithm 2 Create input mask matrix

Input: Generation order $\pi(\cdot)$, constants C_{in}, k_1, k_2, dilation d, is this the first layer?
Start with an empty set of generated coordinates
Initialize $\mathcal{M}$ as $k_1 * k_2 \times H * W$ zero matrix
for i from 1 to $H * W$ **do**
  Let (r, c) be coordinates of dimension $\pi(i)$
  for offsets Δ_r, Δ_c in $k_1 \times k_2$ kernel **do**
    if $(r + d\Delta_r, c + d\Delta_c)$ has been generated **then**
      Allow output location (r, c) to access features at $(r + d\Delta_r, c + d\Delta_c)$ in previous layer: set $\mathcal{M}_{k_2\Delta_r + \Delta_c, Wr+c} = 1$
    end if
  end for
  Add (r, c) to generated coordinates
end for
if not the first layer **then**
  Allow previous layer features to be observed at all locations: set center row $\lfloor \frac{k_1 * k_2}{2} \rfloor$ of $\mathcal{M}$ to 1
end if
Repeat rows of $\mathcal{M}$, C_{in} times
return binary mask matrix $\mathcal{M}$

applied. The inputs to the convolution at each location, *i.e.* the input patches, form columns of X. For a given generation order, we construct mask matrix $\mathcal{M}$ of the same dimensions as X and set $X = \mathcal{M} \odot X$ prior to matrix multiplication. In particular, our locally masked convolution masks *patches of the input to each layer*, rather than masking weights and rather than masking the initial input to the network. LMConv combines the flexibility of NADE and the parallelizability of MADE and PixelCNN. The LMConv algorithm is summarized in Algorithm 1, and mask construction is detailed in Algorithm 2.

We implement two versions of the layer with the PyTorch machine learning framework (Paszke et al., 2019). The first is an implementation that uses autodifferentiation to compute gradients. As only the forward pass is defined by the user, the implementation is under 20 lines of Python.

However, reverse-mode autodifferentiation incurs significant memory overheads during backpropagation as the output of nearly every operation during the forward pass must be stored until gradient computation (Griewank and Walther, 2000; Jain et al., 2020). Data rearrangement with im2col is memory intensive as features patches overlap and are duplicated. We implement a custom, memory efficient backward pass that only stores the input, the mask and the output of the layer during the forward pass and recomputes the im2col operation during the backward pass. Recomputing the im2col operation achieves 2.7× memory savings at a 1.3× slowdown.

Using locally masked convolutions, we can experiment with many different image generation orders. In this work, we consider three classes of orderings: raster scan, implemented in baseline PixelCNNs, an S-curve order that traverses rows in alternating directions, and a Hilbert space-filling curve order that generates nearby pixels in the image consecutively. Alternate orderings provide several benefits. Nearby pixels in an image are highly correlated. By generating these pixels close in a Hilbert curve order, we might expect information to propagate from the most important, nearby observations for each dimension and reduce the vanishing gradient problem.

If the image is considered a graph with nodes for each pixel and edges connecting adjacent pixels, a convolutional autoregressive model using an order defined by a Hamiltonian path over the image graph will also suffer no blind spot in a D layer network. To see this, note that the features corresponding to dimension $x_{\pi(i)}$ in the Hamiltonian path order will always be able to observe the previous layer's features corresponding to $x_{\pi(i-1)}$. After at least D layers of depth, the features for $x_{\pi(i)}$ will incorporate information from all $i - 1$ previous dimensions. In practice, information propagates with fewer required layers in these architectures as multiple neighbors are observed in each layer. Finally, we select multiple orderings at inference and average the resulting joint distributions to compute better likelihood estimates.

5 ARCHITECTURE

We use a network architecture similar to PixelCNN++ (Salimans et al., 2017), the best-in-class density estimator in the fully convolutional autoregressive PixelCNN model family. Convolution operations are masked according to Algorithm 1. While our locally masked convolutions can benefit from self-attention mechanisms used in later work, we choose a fully convolutional architecture for simplicity and to study the benefit of local masking in isolation of other architectural innovations. We make three modifications to the PixelCNN++ architecture that simplify it and allow for arbitrary generation orders. Gated PixelCNN

Table 1: Average negative log likelihood of binarized and grayscale MNIST digits under our model. Lower is better.

BINARIZED MNIST, 28x28	**NLL (nats)**
DARN (Intractable) (Gregor et al., 2014)	≈84.13
NADE (Uria et al., 2014)	88.33
EoNADE 2hl (128 orders) (Uria et al., 2014)	85.10
EoNADE-5 2hl (128 orders) (Raiko et al., 2014)	84.68
MADE 2hl (32 orders) Germain et al. (2015)	86.64
PixelCNN (van den Oord et al., 2016b)	81.30
PixelRNN (van den Oord et al., 2016b)	79.20
Ours, S-curve (1 order)	78.47
Ours, S-curve (8 orders)	**77.58**
GRAYSCALE MNIST, 28x28	**NLL (bpd)**
Spatial PixelCNN (Akoury and Nguyen, 2017)	0.88
PixelCNN++ (1 stream)	0.77
Ours, S-curve (1 order)	0.68
Ours, S-curve (8 orders)	**0.65**

Table 2: Average negative log likelihood of CIFAR10 images under our model. Lower is better.

CIFAR10, 32x32	**NLL (bpd)**
Uniform Distribution	8.00
Multivariate Gaussian (van den Oord et al., 2016b)	4.70
Attention-based	
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2018)	2.85
Sparse Transformer (Child et al., 2019)	**2.80**
Convolutional	
PixelCNN (1 stream) (van den Oord et al., 2016b)	3.14
Gated PixelCNN (2 stream) (van den Oord et al., 2016a)	3.03
PixelCNN++ (1 stream)	2.99
PixelCNN++ (2 stream) (Salimans et al., 2017)	2.92
Ours, S-curve (1 stream, 1 order)	2.91
Ours, S-curve (1 stream, 8 orders)	**2.89**

uses a two-stream architecture composed of two network stacks with $\lfloor \frac{k}{2} \rfloor \times 1$ and $\lfloor \frac{k}{2} \rfloor \times k$ convolutions to enforce the raster scan order. In the horizontal stream, Gated PixelCNN applies non-square convolutions and feature map shifts or pads to extract information within the same row, to the left of the current dimension. In the vertical stream, Gated PixelCNN extracts information from above. Skip connections between streams allow information to propagate. PixelCNN++ uses a similar architecture based on a U-Net (Ronneberger et al., 2015) with approximately 54M parameters. We replace the two streams with a simple, single stream with the same depth, using LMConv to maintain the autoregressive property. Masks for these convolutions are computed and cached at the beginning of training. Due to the regularizing effect of order-agnostic training, we do not use dropout.

Second, we use dilated convolutions (Yu and Koltun, 2015) at regular intervals in the model rather than downsampling the feature map. Downsampling precludes many orders, as the operation aggregates information from contiguous squares of pixels together without a mask. Dilated convolutions expand the receptive field without limiting the order, as local masks can be customized to hide or reveal specific features accessed by the filter.

Finally, we normalize the feature map across the channel dimension (Li et al., 2019). Normalization allows masks to have varying numbers of ones at each spatial location by rescaling features to the same scale.

As in PixelCNN++, our model represents each conditional with a mixture of 10 discretized logistic distributions that imposes a distribution over binned pixel intensities. For the binarized MNIST dataset (Salakhutdinov and Murray, 2008), we instead use a softmax over two logits. We train with 8 variants of an S-curve (zig-zag) order that traverses each row of the image in alternating directions so that consecutively generated pixels are adjacent, and so that locally masked CNNs with sufficient depth can achieve the maximum allowed receptive field.

Across all quantitative experiments, we use a model with approximately 46M parameters, trained with the Adam optimizer with a learning rate of $2 * 10^{-4}$ decayed by a factor of $1 - 5 * 10^{-6}$ per iteration with clipped gradients. For CelebA-HQ qualitative results, we increase filter count and train a model with 184M parameters. More details are provided in the appendix.

6 EXPERIMENTS

To evaluate the benefits of our approach, we study three scientific questions: (1) *do locally masked autoregressive ensembles estimate more accurate likelihoods on image datasets than single-order models?*, (2) *can the model generalize to novel orders?* and (3) *how important is order selection for image completion?*

We estimate the distribution of three image datasets: 28×28 grayscale and binary (Salakhutdinov and Murray, 2008) MNIST digits, 32×32 8-bit color CIFAR10 natural images, and high-resolution CelebA-HQ 5-bit color face photographs (Karras et al., 2018). Unlike classification, density estimation remains challenging on these datasets. We train the CelebA-HQ models at 256×256 resolution to compare with prior density estimation work, and at a bilinearly downsampled 64×64 resolution.

Our locally masked model achieves better likelihoods than PixelCNN++ by using multiple generation orders. We then show that the model can generalize to generation

Table 3: Average conditional negative log likelihood for **T**op, **L**eft and **B**ottom half image completion.

BINARIZED MNIST 28x28 (nats)	**T**	**L**	**B**
Ours (adversarial order)	41.76	39.83	43.35
Ours (1 max context order)	34.99	32.47	36.57
Ours (2 max context orders)	**34.82**	**32.25**	**36.36**
CIFAR10 32x32 (bpd)	**T**	**L**	**B**
PixelCNN++, 1 stream	3.07	3.10	3.05
PixelCNN++, 2 stream	2.97	2.98	2.93
Ours (1 stream, adversarial order)	2.93	2.98	3.05
Ours (1 stream, 1 max context order)	2.77	2.83	2.89
Ours (1 stream, 2 max context orders)	**2.76**	**2.82**	**2.88**

orders that it has not been trained with. Finally, for image completion, we achieve the best results over strong baselines by using orders that expose all observed pixels.

6.1 WHOLE-IMAGE DENSITY ESTIMATION

Tractable generative models are generally evaluated via the average negative log likelihood (NLL) of test data. For interpretability, many papers normalize base 2 NLL by the number of dimensions. By normalizing, we can measure bits per dimension (bpd), or a lower-bound for the expected number of bits needed per pixel to losslessly compress images using a Huffman code with $p(\mathbf{x})$ estimated by our model. Better estimates of the distribution should result in higher compression rates. Tables 1 and 2 show likelihoods for our model and prior models.

On binarized MNIST (Table 1), our locally masked PixelCNN achieves significantly higher likelihoods (lower NLL) than baselines, including neural autoregressive models NADE, EoNADE, and MADE that average across large numbers of orderings. This is due to architectural advantages of our CNN and increased model capacity. Our model also outperforms the standard PixelCNN, which suffers from a blind spot problem due to sharing the same mask at all locations. Likelihood is further improved by using ensemble averaging across 8 orders that share parameters. These results are also observed on grayscale MNIST where each pixel has one of 256 intensity levels.

On CIFAR10, we achieve 2.89 bpd test set likelihood when averaging the joint probability of 8 graphical models, each defined by an S-curve generation order. Our results outperform the state-of-the-art convolutional autoregressive model, PixelCNN++. We significantly outperform a 1 stream architectural variant of PixelCNN++ that has the same number of parameters as our model and uses a similar architecture, differing only in that it uses a single raster scan order. By introducing order-agnostic ensemble averaging to convolutional autoregressive models, we combined the best of fully-connected density estimators that average over orders, and the inductive biases of CNNs. These results could further improve with self-attention mechanisms and additional capacity, which have been observed to improve the performance of singe-order estimation, marking an opportunity for future research.

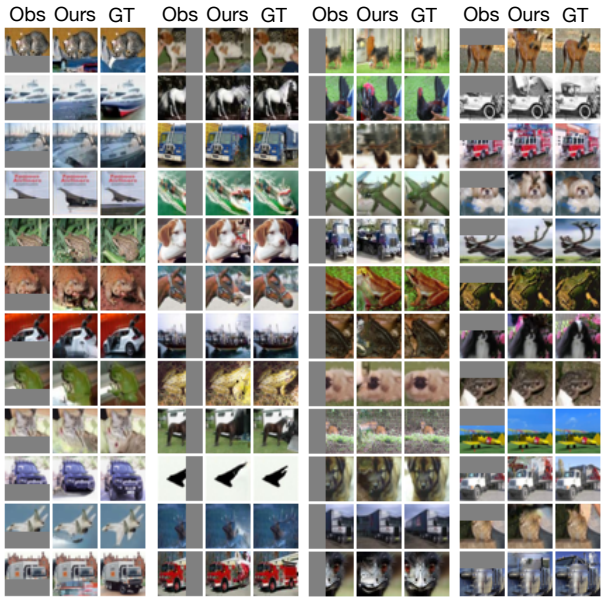


Figure 5: CIFAR10 image completions using our locally-masked convolutions with a specialized ordering.

Our model is also scalable to high resolution distribution estimation. On the CelebA-HQ 256x256 dataset at 5-bit color depth, our model achieves 0.74 bpd with a single S-curve order, outperforming Glow (Kingma and Dhariwal, 2018), an exact likelihood normalizing flow. In comparison, the state-of-the-art model, SPN (Menick and Kalchbrenner, 2019), achieves 0.61 bpd by using self-attention and a specialized architecture for high resolutions.

6.2 GENERALIZATION TO NOVEL ORDERS

Ideally, an order-agnostic model would be able to generate images in orders that it has not been trained with. To understand generalization to novel orders, we evaluate the test-set likelihood of a CIFAR10 model that achieves 2.93 bpd with a single S-curve order and 2.91 bpd with 8 S-curve orders under a raster scan decomposition. The model achieves 3.75 bpd with 1 raster scan order (28% increase) and 3.67 bpd with 8 raster scan orders (26% increase). While the novel order degrades compression rate, the model was trained with 8 fixed orders of the same S-curve type, which are fairly different from a raster scan.

To study generalization to more similar orders, we trained

Figure 6: Completions of 64×64 px CelebA-HQ images at 5-bit color depth. Up to 2 samples are shown to the right of each half-obscured face provided to the model. Missing pixels are generated along an S-curve that first traverses the observed region. Additional samples and ground truth completions are provided in the appendix.

a model on Binarized MNIST with 7 S-curves for 120 epochs. On the test set, the model has 0.144 bpd using each train order. Testing with the held out (8th) S-curve, the model achieves 0.151 bpd, only 5% higher.

6.3 IMAGE COMPLETION

To quantitatively assess whether control over generation order improves image completions, we measure the average conditional negative log likelihood of hidden regions of held-out test images on the MNIST and CIFAR10 datasets, measured in bits per dimension. We compute the NLL of the top half, left half, and bottom half of the image conditioned on the remainder of the image. The hidden region is set to zero in the model input, as well as hidden via masks used in each model.

Table 3 shows average NLL on binary MNIST and CIFAR10. Top half inpainting is challenging for PixelCNN baselines that use a raster scan order, as model conditional $p_\theta(x_i|x_{<i})$ does not condition on observed pixels that lie below x_i in the image. Similarly, our architecture under an adversarial order, a single S-shaped curve from the top left to bottom left of the image, achieves 2.93 bpd on CIFAR in the **T** setting. In contrast, using the same parameters, when we decomposes the joint favorably for maximum context with an S-curve generation order from the bottom left to the top left of the image, we achieve 2.77 bpd. Averaging over two maximum context orders further improves log likelihood to 2.76 bpd. A similar trend is observed for the other completion tasks, **L** and **B**.

6.4 QUALITATIVE RESULTS

Figure 1 shows completions of MNIST and CelebA-HQ 64×64 images. PixelCNN++ produces MNIST digits that are inconsistent with the observed context. With a poor choice of order, our model only respects some attributes of the input image, but not overall facial structure. The model distributions over each missing pixel should condition on the entire observed region. This is accomplished when the missing region is generated last via a maximum context order. With this order, completions by our model are consistent with the given context.

Figures 5 and 6 show completions of held-out CIFAR10 32×32 and CelebA-HQ 64×64 images for four different missing regions. The masked input to the model (Obs), our sampled completion (Ours) and the ground truth image (GT) are shown. Missing image regions are generated in a maximum context order. While samples have some artifacts such as blurring due to long sequence lengths, images are globally coherent, with matching colors and object structure (CIFAR10) or facial structure (CelebA-HQ). Across datasets and image masks, our model effectively uses available context to generate coherent samples.

7 RELATED WORK

Autoregressive models are a popular choice to estimate the joint distribution of high-dimensional, multivariate data in deep learning. Frey (1998) proposes logistic autoregressive Bayesian networks where each conditional is learned through logistic regression, capturing first-order dependencies between variables. While different orders had similar performance, averaging densities from 10 differently ordered models achieved small improvements in likelihood. Bengio and Bengio (2000) extend this idea, using artificial neural networks to capture conditionals with some parameter sharing. Larochelle and Murray (2011) propose the neural autoregressive distribution estimator (NADE) for binary and discrete data, reducing the complexity of density estimation from quadratic in the number of dimensions to linear. Uria et al. (2013) ex-

tend NADE to real-valued vectors (RNADE), expressing conditionals as mixture density networks. The autoregressive approach is desirable due to the lack of conditional independence assumptions, easy training via maximum likelihood, tractable density, and tractable, though sequential, forward sampling directly from the conditionals.

These works all use a single, arbitrary order per estimated model. However, it is possible to use the same parameters to define a family of differently ordered autoregressive Bayesian networks. Uria et al. (2014) propose EoNADE, an ensemble of input-masked NADE models trained with an order-agnostic training procedure that achieve higher likelihoods when averaged and allows forward sampling of arbitrary regions. Each iteration, EoNADE chooses a random prefix of an ordering $\pi(1), \ldots, \pi(d)$, sample a training example x and maximize the likelihood of x_d under their model. ConvNADE (Uria et al., 2016) adapts EoNADE with a convolutional architecture and conditions the model on the input mask defining the order. Still, NADE, EoNADE and ConvNADE are serial: only a single conditional is trained at a time, and density estimation requires D passes. Germain et al. (2015) propose an order-agnostic MADE that masks the weights of a fully connected autoencoder to estimate densities with a single forward pass by computing conditionals in parallel. While MADE supports multiple orders, it is limited by a fully-connected architecture. Our Locally Masked PixelCNN can be seen as a generalization of MADE that supports convolutional inductive bias.

Other deep autoregressive models use recurrent, convolutional or self-attention architectures. In language modeling, autoregressive recurrent neural networks (RNNs) predict a distribution over the next token in a sequence conditioned on a recurrently updated representation of the previous words (Mikolov et al., 2010). van den Oord et al. (2016b) extend this idea to images, proposing a multi-dimensional, sequential PixelRNN for image generation and discrete distribution estimation, and a parallelizable PixelCNN. Subsequent works capture correlations between pixels in an image with convolutional architectures inspired by the PixelCNN (van den Oord et al., 2016a; Salimans et al., 2017; Menick and Kalchbrenner, 2019; Reed et al., 2017), often improving the ability of the network to capture long-range dependencies. The PixelCNN family can generate entire high-fidelity images and, until recently, achieved state-of-the-art test set likelihood among tractable, likelihood-based generative models. PixelCNNs have also been used as a prior for latent variables (van den Oord et al., 2017), and can be sampled in parallel using fixed-point methods (Song et al., 2020; Wiggers and Hoogeboom, 2020). While convolutions process information locally in an image, self-attention mechanisms have been used to gain global receptive field (Chen et al., 2018; Parmar et al., 2018; Child et al., 2019) for improved statistical performance.

Normalizing flows (Rezende and Mohamed, 2015) are parametric density estimators that give exact expressions for likelihood using the change-of-variables formula by transforming samples from a simple prior with learned, invertible functions. If tractable densities are not required, other families are possible. Implicit generative models such as GANs (Goodfellow et al., 2014) have been applied to high resolution image generation (Karras et al., 2018) and inpainting (Pathak et al., 2016). Nonparametric approaches have also been successful for inpainting (Efros and Leung, 1999; Hays and Efros, 2007; Barnes et al., 2009). Partial convolutions (Liu et al., 2018) improve CNN inpainting quality by rescaling filter responses that access missing pixels, but are not causal unlike LM-CONV. Latent-variable models like the VAE (Kingma and Welling, 2014; Rezende et al., 2014) jointly learn a generative model for data x given latent z and an approximation for the posterior over z. Other latent-variable models are based on Markov chains (Bengio et al., 2014; Sohl-Dickstein et al., 2015; Nijkamp et al., 2019).

8 CONCLUSION

In this work, we proposed an efficient, scalable and easy to implement approach for supporting arbitrary autoregressive orderings within convolutional networks. To do so, we propose *locally masked convolutions* that allow arbitrary orderings by masking features at each layer while simultaneously sharing filter weights. This formulation can be efficiently implemented purely via matrix multiplication. Our work is a synthesis of prior lines of inquiry in autoregressive models. Locally Masked PixelCNNs support parallel estimation, convolutional inductive biases, and control over order, all with one simple layer. Foundational work in this area each supported some of these, but with incompatible architectures. As an additional benefit, arbitrary orderings allow image completion with diverse regions. We achieve globally coherent image completions by choosing a favorable order at test time, without specifically training the model to inpaint.

Acknowledgements

We thank Paras Jain, Nilesh Tripuraneni, Joseph Gonzalez and Jonathan Ho for helpful discussions, and reviewers for helpful suggestions. This research is supported in part by the NSF GRFP under grant number DGE-1752814, Berkeley Deep Drive and the Open Philanthropy Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

N. Akoury and A. Nguyen. Spatial PixelCNN: Generating images from patches. *arXiv:1712.00714*, 2017.

C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM TOG*, 28(3), Aug. 2009.

Y. Bengio and S. Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *NIPS*, 2000.

Y. Bengio, E. Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *ICML*, 2014.

J. Červený. Generalized Hilbert space-filling curve, Oct 2019.

X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. PixelSNAIL: An improved autoregressive generative model. 2018.

R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv*, 2019.

Y. Du and I. Mordatch. Implicit generation and modeling with energy based models. *NeurIPS*, 2019.

A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

B. J. Frey. *Graphical models for machine learning and digital communication*. MIT Press, 1998.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: Masked autoencoder for distribution estimation. In *ICML*, pages 881–889, 2015.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *NIPS*, 2014.

K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. Deep autoregressive networks. In *ICML*, volume 32, 2014.

A. Griewank and A. Walther. Algorithm 799: revolve. *ACM TOMS*, 2000.

J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM TOG (SIGGRAPH)*, 26(3), 2007.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8), 2002.

P. Jain, A. Jain, A. Nrusimha, A. Gholami, P. Abbeel, K. Keutzer, I. Stoica, and J. E. Gonzalez. Checkmate: Breaking the memory wall with optimal tensor rematerialization. 2020.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, volume 1, 2014.

D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12(4):307–392, 2019.

H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *AISTATS*, pages 29–37, 2011.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. 2006.

B. Li, F. Wu, K. Q. Weinberger, and S. Belongie. Positional normalization. In *NeurIPS*, 2019.

G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018.

J. Menick and N. Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *ICLR*, 2019.

T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 26–30 Sep 2010.

E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *NeurIPS*, pages 5232–5242, 2019.

N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In *ICML*, 2018.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

K. Popat and R. W. Picard. Novel cluster-based probability model for texture synthesis, classification, and compression. In *VCIP*, volume 2094, pages 756–768, 1993.

T. Raiko, Y. Li, K. Cho, and Y. Bengio. Iterative neural autoregressive distribution estimator NADE-k. In *NIPS*, 2014.

S. Reed, A. van den Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. de Freitas. Parallel multiscale autoregressive density estimation. In *ICML*, 2017.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 22–24 Jun 2014.

O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. 2015.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *ICML*, 2008.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pages 2256–2265, 2015.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

Y. Song, C. Meng, R. Liao, and S. Ermon. Nonlinear equation solving: A faster alternative to feedforward computation, 2020.

B. Uria, I. Murray, and H. Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *NIPS*, 2013.

B. Uria, I. Murray, and H. Larochelle. A deep and tractable density estimator. In *ICML*, 2014.

B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *JMLR*, 2016.

A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with PixelCNN decoders. In *NeurIPS*, 2016a.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016b.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *NIPS*, 2017.

A. J. Wiggers and E. Hoogeboom. Predictive sampling with forecasting autoregressive models, 2020.

F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2015.