MaCow: Masked Convolutional Generative Flow

Xuezhe Ma, Xiang Kong, Shanghang Zhang, Eduard Hovy
Carnegie Mellon University
Pittsburgh, PA, USA
xuezhem,xiangk@cs.cmu.edu, shanghaz@andrew.cmu.edu, hovy@cmu.edu

Abstract

Flow-based generative models, conceptually attractive due to tractability of the exact log-likelihood computation and latent-variable inference as well as efficiency in training and sampling, has led to a number of impressive empirical successes and spawned many advanced variants and theoretical investigations. Despite computational efficiency, the density estimation performance of flow-based generative models significantly falls behind those of state-of-the-art autoregressive models. In this work, we introduce *masked convolutional generative flow* (**MaCow**), a simple yet effective architecture for generative flow using masked convolution. By restricting the local connectivity to a small kernel, MaCow features fast and stable training along with efficient sampling while achieving significant improvements over Glow for density estimation on standard image benchmarks, considerably narrowing the gap with autoregressive models.

1 Introduction

Unsupervised learning of probabilistic models is a central yet challenging problem. Deep generative models have shown promising results in modeling complex distributions such as natural images (Radford et al., 2015), audio (Van Den Oord et al., 2016) and text (Bowman et al., 2015). Multiple approaches emerged in recent years, including Variational Autoencoders (VAEs) (Kingma and Welling, 2014), Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), autoregressive neural networks (Larochelle and Murray, 2011; Oord et al., 2016), and flow-based generative models (Dinh et al., 2014, 2016; Kingma and Dhariwal, 2018). Among these, flow-based generative models gained popularity for this capability of estimating densities of complex distributions, efficiently generating high-fidelity syntheses, and automatically learning useful latent spaces.

Flow-based generative models typically warp a simple distribution into a complex one by mapping points from the simple distribution to the complex data distribution through a chain of invertible transformations with Jacobian determinants that are efficient to compute. This design guarantees that the density of the transformed distribution can be analytically estimated, making maximum likelihood learning feasible. Flow-based generative models have spawned significant interests for improving and analyzing its algorithms both theoretically and practically, and applying them to a wide range of tasks and domains.

In their pioneering work, Dinh et al. (2014) first proposed *Non-linear Independent Component Estimation* (NICE) to apply flow-based models for modeling complex high-dimensional densities. RealNVP (Dinh et al., 2016) extended NICE with a more flexible invertible transformation to experiment with natural images. However, these flow-based generative models resulted in worse density estimation performance compared to state-of-the-art autoregressive models, and are incapable of realistic synthesis of large images compared to GANs (Karras et al., 2018; Brock et al., 2019). Recently, Kingma and Dhariwal (2018) proposed Glow as a generative flow with invertible 1×1 convolutions, which significantly improved the density estimation performance on natural images. Importantly, they demonstrated that flow-based generative models optimized towards the plain

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

likelihood-based objective are capable of generating realistic high-resolution natural images efficiently. Prenger et al. (2018) investigated applying flow-based generative models to speech synthesis by combining Glow with WaveNet (Van Den Oord et al., 2016). Ziegler and Rush (2019) adopted variational inference to apply generative flows to discrete sequential data. Unfortunately, the density estimation performance of Glow on natural images remains behind autoregressive models, such as PixelRNN/CNN (Oord et al., 2016; Salimans et al., 2017), Image Transformer (Parmar et al., 2018), PixelSNAIL (Chen et al., 2017) and SPN (Menick and Kalchbrenner, 2019). There is also some work (Rezende and Mohamed, 2015; Kingma et al., 2016; Zheng et al., 2017) trying to apply flow to variational inference.

In this paper, we propose a novel architecture of generative flow, *masked convolutional generative flow* (**MACOW**), which leverages masked convolutional neural networks (Oord et al., 2016). The bijective mapping between input and output variables is easily established while the computation of the determinant of the Jacobian remians efficient. Compared to inverse autoregressive flow (IAF) (Kingma et al., 2016), MACOW offers stable training and efficient inference and synthesis by restricting the local connectivity in a small "masked" kernel as well as large receptive fields by stacking multiple layers of convolutional flows and using rotational ordering masks (§3.1). We also propose a fine-grained version of the multi-scale architecture adopted in previous flow-based generative models to further improve the performance (§3.2). Experimenting with four benchmark datasets for images, CIFAR-10, ImageNet, LSUN, and CelebA-HQ, we demonstrate the effectiveness of MACOW as a density estimator by consistently achieving significant improvements over Glow on all the three datasets. When equipped with the variational dequantization mechanism (Ho et al., 2019), MACOW considerably narrows the gap of the density estimation with autoregressive models (§4).

2 Flow-based Generative Models

In this section, we first setup notations, describe flow-based generative models, and review Glow (Kingma and Dhariwal, 2018) as it is the foundation for MACOW.

2.1 Notations

Throughout the paper, uppercase letters represent random variables and lowercase letters for realizations of their corresponding random variables. Let $X \in \mathcal{X}$ be the random variables of the observed data, e.g., X is an image or a sentence for image and text generation, respectively.

Let P denote the true distribution of the data, i.e., $X \sim P$, and $D = \{x_1, \ldots, x_N\}$ be our training sample, where $x_i, i = 1, \ldots, N$, are usually i.i.d. samples of X. Let $\mathcal{P} = \{P_\theta : \theta \in \Theta\}$ denote a parametric statistical model indexed by the parameter $\theta \in \Theta$, where Θ is the parameter space. p denotes the density of the corresponding distribution P. In the deep generative model literature, deep neural networks are the most widely used parametric models. The goal of generative models is to learn the parameter θ such that P_θ can best approximate the true distribution P. In the context of maximum likelihood estimation, we minimize the negative log-likelihood of the parameters with:

$$\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} -\log p_\theta(x_i) = \min_{\theta \in \Theta} \mathrm{E}_{\widetilde{P}(X)}[-\log p_\theta(X)], \tag{1}$$

where $\tilde{P}(X)$ is the empirical distribution derived from training data D.

2.2 Flow-based Models

In the framework of flow-based generative models, a set of latent variables $Z \in \mathcal{Z}$ are introduced with a prior distribution $p_Z(z)$, which is typically a simple distribution like a multivariate Gaussian. For a bijection function $f : \mathcal{X} \to \mathcal{Z}$ (with $g = f^{-1}$), the change of the variable formula defines the model distribution on X by

$$p_\theta(x) = p_Z\left(f_\theta(x)\right) \left| \det\left(\frac{\partial f_\theta(x)}{\partial x}\right) \right|, \tag{2}$$

where $\frac{\partial f_\theta(x)}{\partial x}$ is the Jacobian of f_θ at x.

The generative process is defined straightforwardly as the following:

$$\begin{array}{rcl} z & \sim & p_Z(z) \\ x & = & g_\theta(z). \end{array} \tag{3}$$

Flow-based generative models focus on certain types of transformations f_θ that allow the inverse functions g_θ and Jacobian determinants to be tractable to compute. By stacking multiple such invertible transformations in a sequence, which is also called a (normalizing) *flow* (Rezende and Mohamed, 2015), the flow is then capable of warping a simple distribution ($p_Z(z)$) into a complex one ($p(x)$) through:

$$\begin{tikzcd} X \arrow[r, leftrightarrow, "f_1", "g_1"'] & H_1 \arrow[r, leftrightarrow, "f_2", "g_2"'] & H2 \arrow[r, leftrightarrow, "f_3", "g_3"'] & \cdots \arrow[r, leftrightarrow, "f_K", "g_K"'] & Z, \end{tikzcd}$$

where $f = f_1 \circ f_2 \circ \cdots \circ f_K$ is a flow of K transformations. For brevity, we omit the parameter θ from f_θ and g_θ.

2.3 Glow

Recently, several types of invertible transformations emerged to enhance the expressiveness of flows, among which Glow (Kingma and Dhariwal, 2018) has stood out for its simplicity and effectiveness on both density estimation and high-fidelity synthesis. The following briefly describes the three types of transformations that comprise Glow.

Actnorm. Kingma and Dhariwal (2018) proposed an activation normalization layer (Actnorm) as an alternative for batch normalization (Ioffe and Szegedy, 2015) to alleviate the challenges in model training. Similar to batch normalization, Actnorm performs an affine transformation of the activations using a scale and bias parameter per channel for 2D images, such that

$$y_{i,j} = s \odot x_{i,j} + b,$$

where both x and y are tensors of shape $[h \times w \times c]$ with spatial dimensions (h, w) and channel dimension c.

Invertible 1×1 convolution. To incorporate a permutation along the channel dimension, Glow includes a trainable invertible 1×1 convolution layer to generalize the permutation operation as:

$$y_{i,j} = W x_{i,j},$$

where W is the weight matrix with shape $c \times c$.

Affine Coupling Layers. Following Dinh et al. (2016), Glow includes affine coupling layers in its architecture of:

$$\begin{array}{rcl} x_a, x_b & = & \mathrm{split}(x) \\ y_a & = & x_a \\ y_b & = & \mathrm{s}(x_a) \odot x_b + \mathrm{b}(x_a) \\ y & = & \mathrm{concat}(y_a, y_b), \end{array}$$

where $\mathrm{s}(x_a)$ and $\mathrm{b}(x_a)$ are outputs of two neural networks with x_a as input. The $\mathrm{split}()$ and $\mathrm{concat}()$ functions perform operations along the channel dimension.

From this designed architecture of Glow, we see that interactions between spatial dimensions are incorporated only in the coupling layers. The coupling layer, however, is typically costly for memory resources, making it infeasible to stack a significant number of coupling layers into a single model, especially when processing high-resolution images. The main goal of this work is to design a new type of transformation that simultaneously models the dependencies in both the spatial and channel dimensions while maintaining a relatively small memory footprint to improve the capacity of the generative flow.

3 Masked Convolutional Generative Flows

In this section, we describe the architectural components of the *masked convolutional generative flow* (MACOW). First, we introduce the proposed flow transformation using masked convolutions in §3.1. Then, we present a fine-grained version of the multi-scale architecture adopted by previous generative flows (Dinh et al., 2016; Kingma and Dhariwal, 2018) in §3.2.

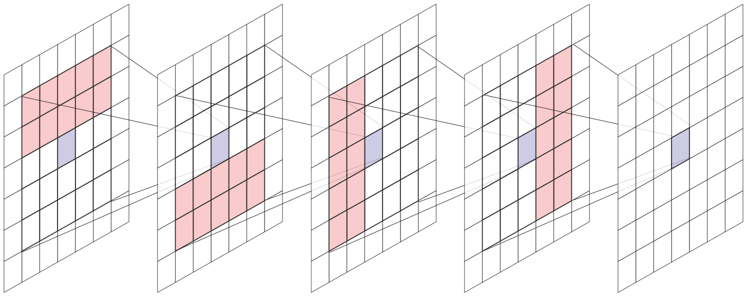


Figure 1: Visualization of the receptive field of four masked convolutions with rotational ordering.

3.1 Flow with Masked Convolutions

Applying autoregressive models to normalizing flows has been previously explored in studies (Kingma et al., 2016; Papamakarios et al., 2017), with idea of sequentially modeling the input random variables in an autoregressive order to ensure the model cannot read input variables behind the current one:

$$y_t = \mathrm{s}(x_{<t}) \odot x_t + \mathrm{b}(x_{<t}), \tag{4}$$

where $x_{<t}$ denotes the input variables in x positioned ahead of x_t in the autoregressive order. $\mathrm{s}()$ and $\mathrm{b}()$ are two autoregressive neural networks typically implemented using spatial masks (Germain et al., 2015; Oord et al., 2016).

Despite effectiveness in high-dimensional space, autoregressive flows suffer from two crucial problems: (1) The training procedure is unstable when stacking multiple layers to increase the flow capacities for complex distributions. (2) Inference and synthesis are inefficient, due to the non-parallelizable inverse function.

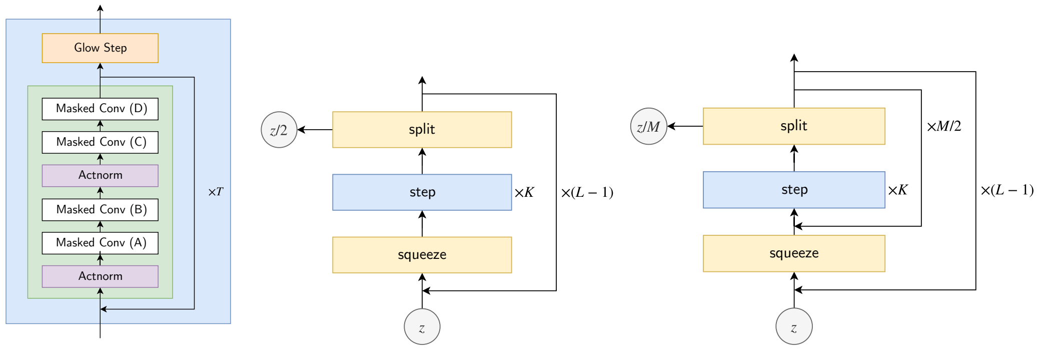
We propose to use masked convolutions to restrict the local connectivity in a small "masked" kernel to address these two problems. The two autoregressive neural networks, $\mathrm{s}()$ and $\mathrm{b}()$, are implemented with one-layer masked convolutional networks with small kernels (e.g. 2×5 in Figure 1) to ensure they only read contexts in a small neighborhood based on:

$$\mathrm{s}(x_{<t}) = \mathrm{s}(x_{t\star}), \quad \mathrm{b}(x_{<t}) = \mathrm{b}(x_{t\star}), \tag{5}$$

where $x_{t\star}$ denotes the input variables, restricted in a small kernel, on which x_t depends. By using masks in rotational ordering and stacking multiple layers of flows, the model captures a large receptive field (see Figure 1), and models dependencies in both the spatial and channel dimensions.

Efficient Synthesis. As discussed above, synthesis from autoregressive flows is inefficient since the inverse must be computed by sequentially traversing through the autoregressive order. In the context of 2D images with shape $[h \times w \times c]$, the time complexity of synthesis is quadratic, i.e. $O(h \times w \times \mathrm{NN}(h, w, c))$, where $\mathrm{NN}(h, w, c)$ is the time of computing the outputs from the neural network $\mathrm{s}()$ and $\mathrm{b}()$ with input shape $[h \times w \times c]$. In our proposed flow with masked convolutions, computation of $x_{i,j}$ begins as soon as all $x_{t\star}$ are available, contrary to the autoregressive requirement that all $x_{<i,j}$ must have been already computed. Moreover, at each step we only need to feed a slice of the image (with shape $[h \times kw \times c]$ or $[kh \times w \times c]$ depending on the direction of the mask) into $\mathrm{s}()$ and $\mathrm{b}()$. Here $[kh \times kw \times c]$ is the shape of the kernel in the convolution. Thus, the time complexity reduces significantly from quadratic to linear, which is $O(h \times \mathrm{NN}(kh, w, c))$ or $O(w \times \mathrm{NN}(kw, h, c))$ for horizontal and vertical masks, respectively.

Discussion The previous work closely related to MACOW is the Emerging Convolutions proposed in Hoogeboom et al. (2019). There are two main differences. i) the pattern of the mask is different. Emerging Convolutions use "causal masks" (Oord et al., 2016) whose inverse function falls into a complete autoregressive transformation. In contrast, MACOW achieves significantly more efficient inference and sampling (§4.3), due to the carefully designed masks (Figure 1). ii) the Emerging Convolutional Flow, proposed as an alternative to the invertible 1×1 convolution in Glow, is basically a linear transformation with masked convolutions, which does not introduce "nonlinearity" to the random variables. MACOW, however, introduces such nonlinearity similar to the coupling layers.



(a) One step of MACOW (b) Original multi-scale architecture (c) Fine-grained multi-scale architecture

Figure 2: The architecture of the proposed MACOW model, where each step (a) consists of T units of *ActNorm* followed by two *masked convolutions* with rotational ordering, and a Glow step. This flow is combined with either an original multi-scale (b) or a fine-grained architecture (c).

3.2 Fine-grained Multi-Scale Architecture

Dinh et al. (2016) proposed a multi-scale architecture using a squeezing operation, which has been demonstrated to be helpful for training very deep flows. In the original multi-scale architecture, the model factors out half of the dimensions at each scale to reduce computational and memory costs. In this paper, inspired by the size upscaling in subscale ordering (Menick and Kalchbrenner, 2019) that generates an image as a sequence of sub-images with equal size, we propose a fine-grained multi-scale architecture to improve model performance further. In this fine-grained multi-scale architecture, each scale consists of $M/2$ blocks, and after each block, the model splits out $1/M$ dimensions of the input[1]. Figure 2 illustrates the graphical specification of the two architecture versions. Note that the fine-grained architecture reduces the number of parameters compared with the original architecture with the same number of flow layers. Experimental improvements demonstrate the effectiveness of the fine-grained multi-scale architecture (§4).

4 Experiments

We evaluate our MACOW model on both low-resolution and high-resolution datasets. For a step of MACOW, we use $T = 2$ masked convolution units, and the Glow step is the same as that described in Kingma and Dhariwal (2018) where an *ActNorm* is followed by an *Invertible* 1×1 *convolution*, which is followed by a *coupling layer*. Each coupling layer includes three convolution layers where the first and last convolutions are 3×3, while the center convolution is 1×1. For low-resolution images, we use affine coupling layers with 512 hidden channels, while for high-resolution images we use additive layers with 256 hidden channels to reduce memory cost. ELU (Clevert et al., 2015) is used as the activation function throughout the flow architecture. For variational dequantization, the dequantization noise distribution $q_\phi(u|x)$ is modeled with a conditional MACOW with shallow architecture. Additional details on architectures, results, and analysis of the conducted experiments are provided in Appendix B.

4.1 Low-Resolution Images

We begin our experiments with an evaluation of the density estimation performance of MACOW on two low-resolution image datasets that are commonly used to evaluate the deep generative models: CIFAR-10 with images of size 32×32 (Krizhevsky and Hinton, 2009) and the 64×64 downsampled version of ImageNet (Oord et al., 2016).

We perform experiments to dissect the effectiveness of each component of our MACOW model with ablation studies. The Org model utilizes the original multi-scale architecture, while the +fine-grained model augments the original one with the fine-grained multi-scale architecture proposed in §3.2. The

[1] In our experiments, we set $M = 4$. Note that the original multi-scale architecture is a special case of the fine-grained version with $M = 2$.

Table 1: Density estimation performance on CIFAR-10 32×32 and ImageNet 64×64. Results are reported in *bits/dim*.

	Model	CIFAR-10	ImageNet-64
Autoregressive	IAF VAE (Kingma et al., 2016)	3.11	–
	Parallel Multiscale (Reed et al., 2017)	–	3.70
	PixelRNN (Oord et al., 2016)	3.00	3.63
	Gated PixelCNN (van den Oord et al., 2016)	3.03	3.57
	MAE (Ma et al., 2019)	2.95	–
	PixelCNN++ (Salimans et al., 2017)	2.92	–
	PixelSNAIL (Chen et al., 2017)	**2.85**	**3.52**
	SPN (Menick and Kalchbrenner, 2019)	–	**3.52**
Flow-based	Real NVP (Dinh et al., 2016)	3.49	3.98
	Glow (Kingma and Dhariwal, 2018)	3.35	3.81
	Flow++: Unif (Ho et al., 2019)	3.29	–
	Flow++: Var (Ho et al., 2019)	**3.09**	3.69
	MaCow: Org	3.31	3.78
	MaCow: +fine-grained	3.28	3.75
	MaCow: +var	3.16	**3.69**

Table 2: Negative log-likelihood scores for 5-bit LSUN and CelebA-HQ datasets in bits/dim.

		LSUN		
Model	**CelebA-HQ**	bedroom	tower	church
Glow (Kingma and Dhariwal, 2018)	1.03	1.20	–	–
SPN (Menick and Kalchbrenner, 2019)	**0.61**	–	–	–
MaCow: Unif	0.95	1.16	1.22	1.36
MaCow: Var	0.67	**0.98**	**1.02**	**1.09**

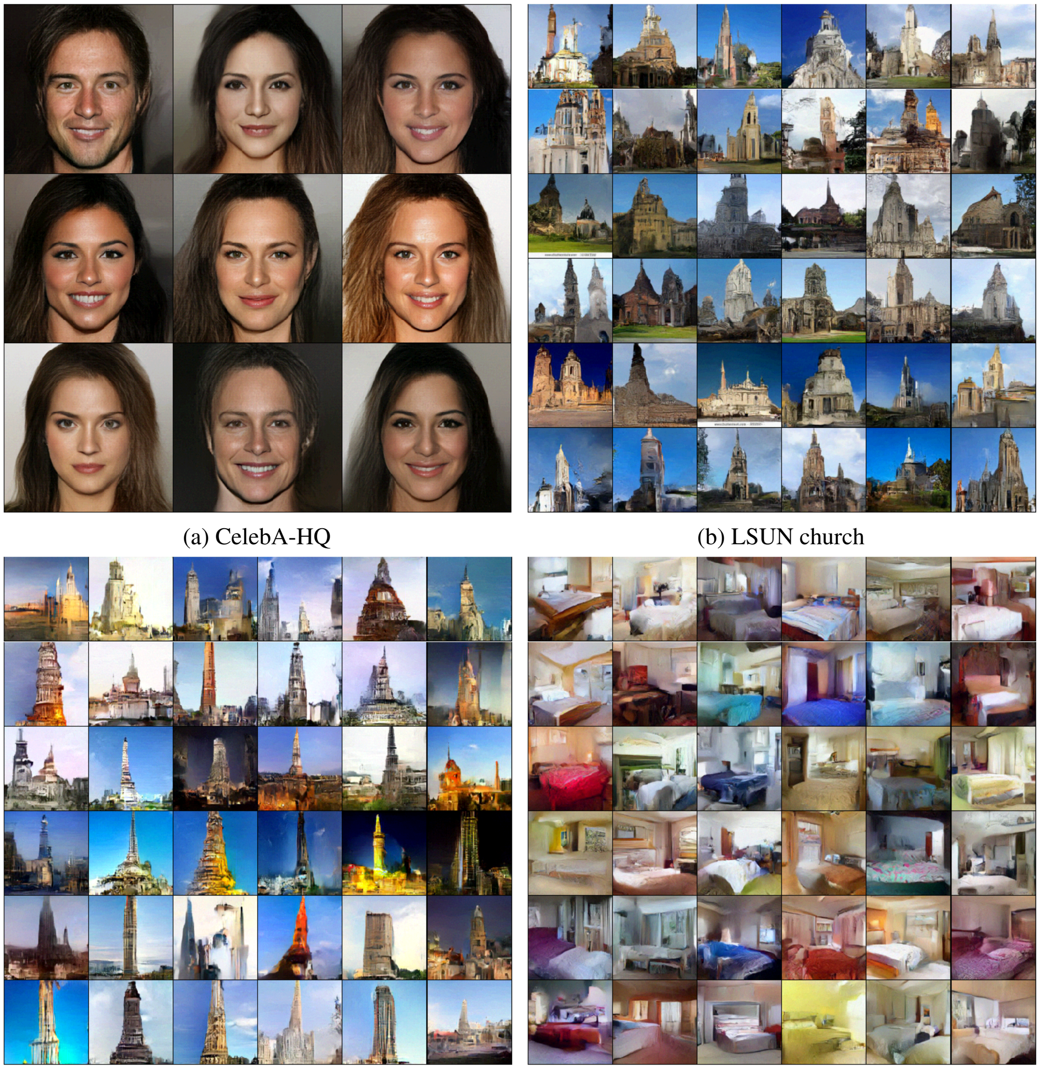
+var model further implements the variational dequantization on the top of +fine-grained to replace the uniform dequantization (see Appendix A for details). For each ablation, we slightly adjust the number of steps in each level so that all the models have a similar number of parameters.

Table 1 provides the density estimation performance for different variations of our MaCow model along with the top-performing autoregressive models (first section) and flow-based generative models (second section). First, on both datasets, fine-grained models outperform Org ones, demonstrating the effectiveness of the fine-grained multi-scale architecture. Second, with the uniform dequantization, MaCow combined with the fine-grained multi-scale architecture significantly improves the performance over Glow on both datasets, and obtains slightly better results than Flow++ on CIFAR-10. In addition, with variational dequantization, MaCow achieves comparable result in bits/dim with Flow++ on ImageNet 64×64. On CIFAR-10, however, the performance of MaCow is around 0.07 bits/dim behind Flow++.

Compared with the state-of-the-art autoregressive generative models PixelSNAIL (Chen et al., 2017) and SPN (Menick and Kalchbrenner, 2019), the performance of MaCow is approximately 0.31 bits/dim worse on CIFAR-10 and 0.14 worse on ImageNet 64×64. Further improving the density estimation performance of MaCow on natural images is left to future work.

4.2 High-Resolution Images

We next demonstrate experimentally that our MaCow model is capable of high fidelity samples at high-resolution. Following Kingma and Dhariwal (2018), we choose the CelebA-HQ dataset (Karras et al., 2018), which consists of 30,000 high-resolution images from the CelebA dataset (Liu et al., 2015), and the LSUN (Yu et al., 2015) datasets including categories *bedroom*, *tower* and *church*. We train our models on 5-bit images with the fine-grained multi-scale architecture and both the uniform and variational dequantization. For each model, we adjust the number of steps in each level so that all the models have similar numbers of parameters with Glow for a fair comparison.



(a) CelebA-HQ

(b) LSUN church

(c) LSUN tower

(d) LSUN bedroom

Figure 3: (a) 5-bit 256 × 256 CelebA-HQ samples with temperature 0.7; (b)(c)(d) 5-bit 128 × 128 LSUN church, tower and bedroom samples, with temperature 0.9, respectively.

4.2.1 Density Estimation

Table 2 illustrates the negative log-likelihood scores in bits/dim of two versions of MaCow on the 5-bit 128 × 128 LSUN and 256 × 256 CelebA-HQ datasets. With uniform dequantization, MaCow improves the log-likelihood over Glow from 1.03 bits/dim to 0.95 bits/dim on CelebA-HQ dataset. Equipped with variational dequantization, MaCow obtains 0.67 bits/dim, which is 0.06 bits/dim behind the state-of-the-art autoregressive generative model SPN (Menick and Kalchbrenner, 2019) and significantly narrows the gap. On the LSUN datasets, MaCow with uniform dequantization outperforms Glow with 0.4 bits/dim on the bedroom category. With variational dequantization, the model achieves further improvements on all the three categories of LSUN datasets,

4.2.2 Image Generation

Consistent with previous work on likelihood-based generative models (Parmar et al., 2018; Kingma and Dhariwal, 2018), we found that sampling from a reduced-temperature model often results in higher-quality samples. Figure 3 showcases some random samples for 5-bit CelebA-HQ 256 × 256 with temperature 0.7 and LSUN 128 × 128 with temperature 0.9. The images are extremely high

Table 3: (a) Image synthesis speed on CIFAR10. Glow re-implemented in PyTorch is masked with †. ‡ denotes results shown in Hoogeboom et al. (2019). (b) Image synthesis speed of MaCow on datasets with different image sizes. The time is measured in milliseconds to sample a datapoint when computed in mini-batchs with size 100.

(a)

CIFAR10	time (ms)	Slow-down
Glow‡	5	1.0
MAF ‡	3000	600.0
Emerging‡	1800	360.0
Glow†	5.3	1.0
MaCow	38.7	7.3

(b)

Dataset	image size	time (ms)
CIFAR10	32×32	38.7
ImageNet	64×64	104.7
LSUN	128×128	267.9
CelebA-HQ	256×256	434.2

quality for non-autoregressive likelihood models, despite that maximum likelihood is a principle that values diversity over sample quality in a limited capacity setting (Theis et al., 2016). More samples of images, including samples of low-resolution ones, are provided in Appendix C[2].

4.3 Comparison on Synthesis Speed

In this section, we compare the synthesis speed of MaCow at test time with that of Glow (Kingma and Dhariwal, 2018), Masked Autoregressive Flows (MAF) (Papamakarios et al., 2017) and Emerging Convolutions (Hoogeboom et al., 2019). Following Hoogeboom et al. (2019), we measure the time to sample a datapoint when computed in mini-batchs with size 100. For fair comparison, we re-implemented Glow using PyTorch (Paszke et al., 2017), and all experiments are conducted on a single NVIDIA TITAN X GPU.

Table 3a shows the sampling speed of MaCow on CIFAR-10, together with that of the baselines. MaCow is 7.3 times slower than Glow, much faster than Emerging Convolution and MAF, whose factors are 360 and 600 respectively. The sampling speed of MaCow on datasets with different image sizes is shown in Table 3b. We see that the time of synthesis increases approximately linearly with the increase of image resolution.

5 Conclusion

In this paper, we propose a new type of generative flow, coined MaCow, which exploits masked convolutional neural networks. By restricting the local dependencies in a small masked kernel, MaCow boasts fast and stable training as well as efficient sampling. Experiments on both low- and high-resolution benchmark datasets of images show the capability of MaCow on both density estimation and high-fidelity generation, achieving state-of-the-art or comparable likelihood as well as its superior quality of samples compared to previous top-performing models[3]

A potential direction for future work is to extend MaCow to other forms of data, in particular text, on which no attempt (to the best of our knowledge) has been made to apply flow-based generative models. Another exciting direction is to combine MaCow with variational inference to automatically learn meaningful (low-dimensional) representations from raw data.

References

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019.

[2] The reduced-temperature sampling is only applied to LSUN and CelebA-HQ 5-bits images, where MaCow adopts additive coupling layers. For CIFAR-10 and ImageNet 8-bits images, we sample with temperature 1.0.

[3] https://github.com/XuezheMax/macow

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. *arXiv preprint arXiv:1712.09763*, 2017.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

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

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS-2014)*, pages 2672–2680, 2014.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730, 2019.

Emiel Hoogeboom, Rianne Van Den Berg, and Max Welling. Emerging convolutions for generative normalizing flows. In *International Conference on Machine Learning*, pages 2771–2780, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2th International Conference on Learning Representations (ICLR-2014)*, Banff, Canada, April 2014.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2011*, pages 29–37, 2011.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015.

Xuezhe Ma, Chunting Zhou, and Eduard Hovy. MAE: Mutual posterior-divergence regularization for variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2019.

Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling, 2019.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of International Conference on Machine Learning (ICML-2016)*, 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, and Alexander Ku. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. *arXiv preprint arXiv:1811.00002*, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Scott Reed, Aäron Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando Freitas. Parallel multiscale autoregressive density estimation. In *International Conference on Machine Learning*, pages 2912–2921, 2017.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Tim Salimans, Andrej Karpathy, Xi Chen, Diederik P Kingma, and Yaroslav Bulatov. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations (ICLR)*, 2017.

L Theis, A van den Oord, and M Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR 2016)*, pages 1–10, 2016.

Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183, 2013.

Aäron 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. 2016.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Guoqing Zheng, Yiming Yang, and Jaime G. Carbonell. Convolutional normalizing flows. *CoRR*, abs/1711.02255, 2017.

Zachary M Ziegler and Alexander M Rush. Latent normalizing flows for discrete sequences. In *Proceedings of International Conference on Machine Learning (ICML-2019)*, 2019.

Appendix: MaCow: Masked Convolutional Generative Flow

A Dequantization

As described in §2, generative flows are defined on continuous random variables. Many real-world datasets, however, are recordings of discrete representations of signals, and fitting a continuous density model to discrete data produces a degenerate solution that places all probability mass on discrete datapoints (Uria et al., 2013; Ho et al., 2019). A common solution to this problem is "dequantization" that converts the discrete data distribution into a continuous one.

Specifically, in the context of natural images, each dimension (pixel) of the discrete data x takes on values in $\{0, 1, \ldots, 255\}$. The dequatization process adds continuous random noise u to x, resulting a continuous data point of:

$$y = x + u, \tag{1}$$

where $u \in [0, 1)^d$ is continuous random noise taking values from interval $[0, 1)$. By modeling the density of $Y \in \mathcal{Y}$ with $p_\theta(y)$, the distribution of X is defined as:

$$P_\theta(x) = \int_{\mathcal{Y}} p_\theta(y)\,\mathrm{d}y = \int_{[0,1)^d} p_\theta(x+u)\,\mathrm{d}u. \tag{2}$$

By restricting the range of u in $[0, 1)$, the mapping between y and a pair of x and u is bijective. Thus, we have $p_\theta(y) = p_\theta(x + u) = p_\theta(x, u)$.

By introducing a *dequantization noise distribution* $q(u|x)$, the training objective in (1) can be re-written as:

$$\begin{aligned}
\mathrm{E}_{P(X)}\Big[-\log P_\theta(X)\Big] &= \mathrm{E}_{P(X)}\left[-\log \int_{[0,1)^d} p_\theta(X,u)\,\mathrm{d}u\right] \\
&= \mathrm{E}_{P(X)}\left[\mathrm{E}_{q(u|X)}\left[-\log \frac{p_\theta(X,u)}{q(u|X)}\right] - \mathrm{KL}\big(q(u|X)||p_\theta(u|X)\big)\right] \\
&\le \mathrm{E}_{P(X)}\left[\mathrm{E}_{q(u|X)}\Big[-\log p_\theta(X,u)\Big] + \mathrm{E}_{q(u|X)}\Big[\log q(u|X)\Big]\right] \\
&= \mathrm{E}_{p(Y)}\Big[-\log p_\theta(Y)\Big] + \mathrm{E}_{P(X)}\mathrm{E}_{q(u|X)}\Big[\log q(u|X)\Big],
\end{aligned} \tag{3}$$

where $p(y) = P(x)q(u|x)$ is the distribution of the dequantized variable Y under the dequantization noise distribution $q(u|X)$.

Uniform Dequantization. The most common dequantization method in prior work is uniform dequantization where the noise u is sampled from the uniform distribution $\mathrm{Unif}(0, 1)$ such that

$$q(u|x) \sim \mathrm{Unif}(0,1), \forall x \in \mathcal{X}.$$

From (3), we have

$$\mathrm{E}_{P(X)}\left[-\log P_\theta(X)\right] \le \mathrm{E}_{p(Y)}\left[-\log p_\theta(Y)\right],$$

as $\log q(u|x) = 0, \forall x \in \mathcal{X}$.

Variational Dequantization. As discussed in Ho et al. (2019), uniform dequantization directs $p_\theta(y)$ to assign uniform density to unit hypercubes $[0, 1)^d$, which is difficult for smooth distribution approximators. They proposed a parametric dequantization noise distribution $q_\phi(u|x)$ with a training objective to optimize the *evidence lower bound* (ELBO) provided in (3):

$$\min_{\theta,\phi} \mathrm{E}_{p_\phi(Y)}\left[-\log p_\theta(Y)\right] + \mathrm{E}_{P(X)}\mathrm{E}_{q_\phi(u|X)}\left[\log q_\phi(u|X)\right], \tag{4}$$

where $p_\phi(y) = P(x)q_\phi(u|x)$. In this paper, we implemented both these two dequantization methods for our MaCow, as is detailed in §4).

B Experimental Details

B.1 Model details

Table 4: Hyper-parameters for MaCow in our experiments.

DataSet	Dequant	Batch Size	Levels	Depths per Level	# Param	# Param Glow
CIFAR-10	Unif	512	3	$[[12,12],[12,12],12]$	41.2M	44.2M
	Var	512	3	$[[12,12],[12,12],12]$	43.5M	
ImageNet	Unif	160	4	$[[16,16],[16,16],[12,12],12]$	117.2M	111.6M
	Var	160	4	$[[16,16],[16,16],[12,12],12]$	123.1M	
LSUN	Unif	160	5	$[[32,32],[32,32],[16,16],[12,12],6]$	166.6M	198.1M
	Var	160	5	$[[32,32],[32,32],[16,16],[12,12],6]$	171.9M	
CelebA-HQ	Unif	40	6	$[[24,24],[16,16],[16,16],[8,8],[4,4],2]$	171.9M	170.8M
	Var	40	6	$[[24,24],[16,16],[16,16],[8,8],[4,4],2]$	177.3M	

B.2 Optimization

Parameter optimization is performed with the Adam optimizer (Kingma and Ba, 2014) with $\beta = (0.9, 0.999)$ and $\epsilon = 1\mathrm{e}-8$. Warmup training is applied to all the experiments: the learning rate linearly increases to for 500 updates to the initial learning rate $1\mathrm{e}-3$. Then we use exponential decay to decrease the learning rate with decay rate is 0.999997.

C More samples from our experiments

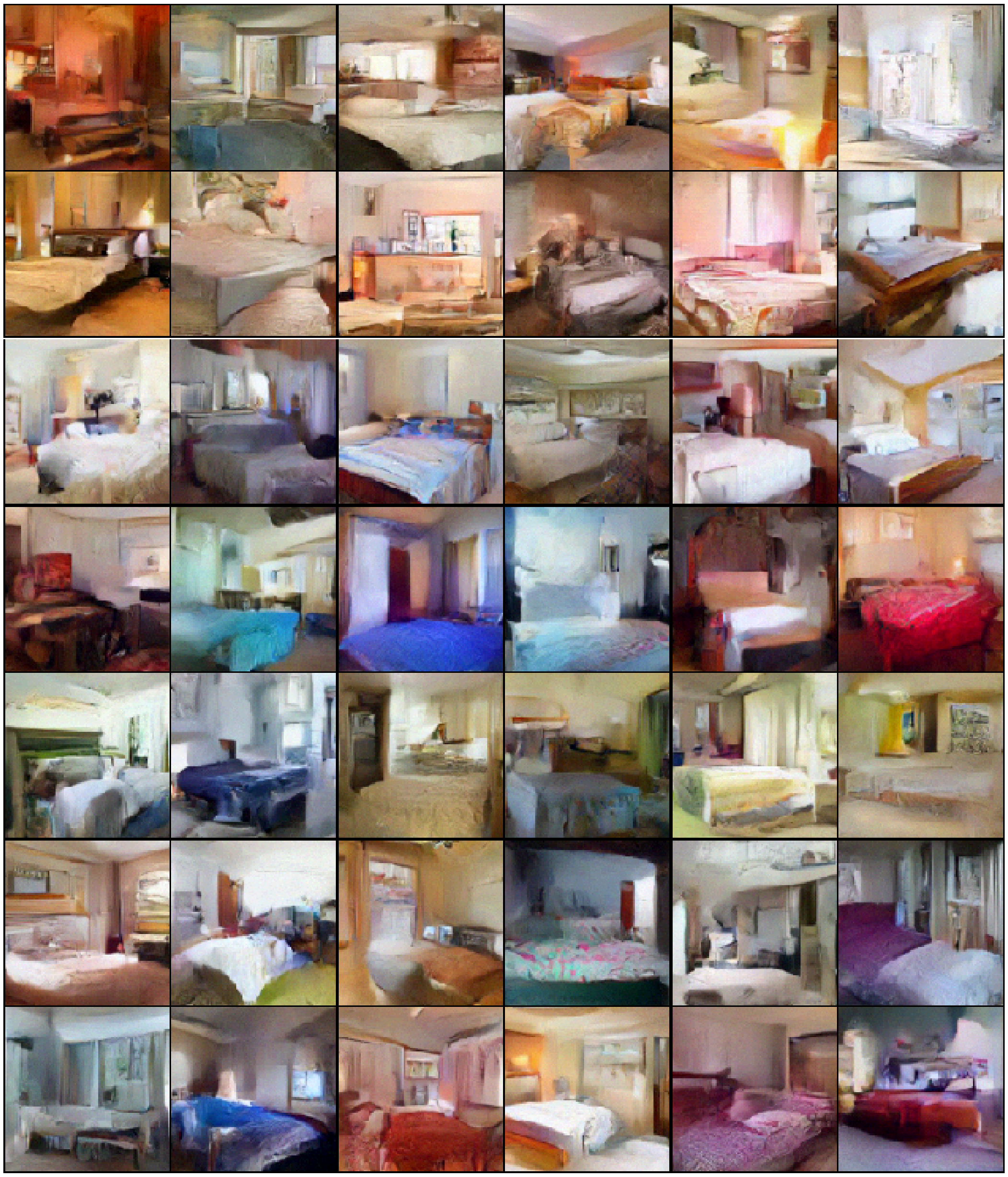


Figure 4: Samples from 5-bit, 128×128 LSUN bedrooms.

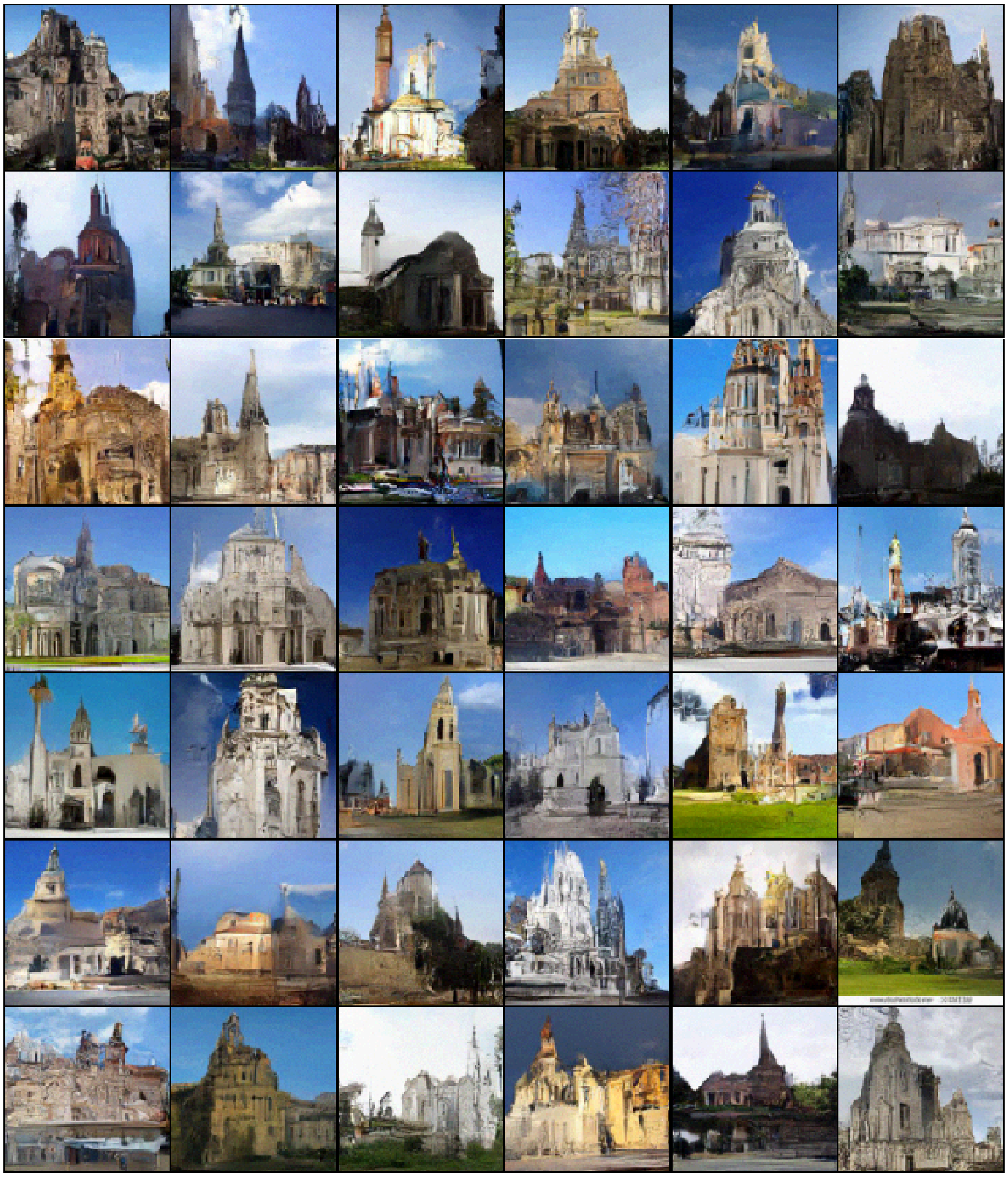


Figure 5: Samples from 5-bit, 128×128 LSUN church.

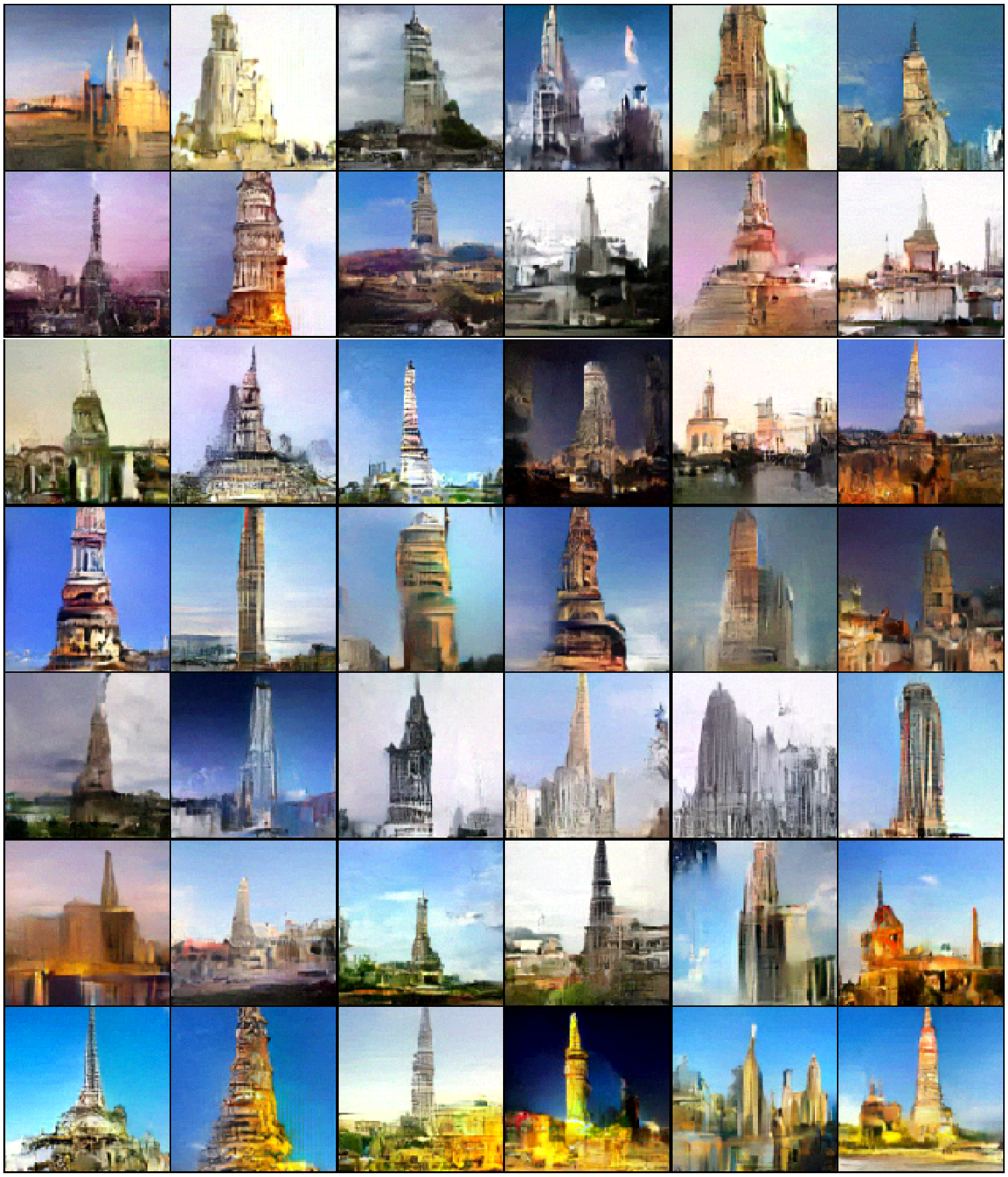


Figure 6: Samples from 5-bit, 128×128 LSUN towers.

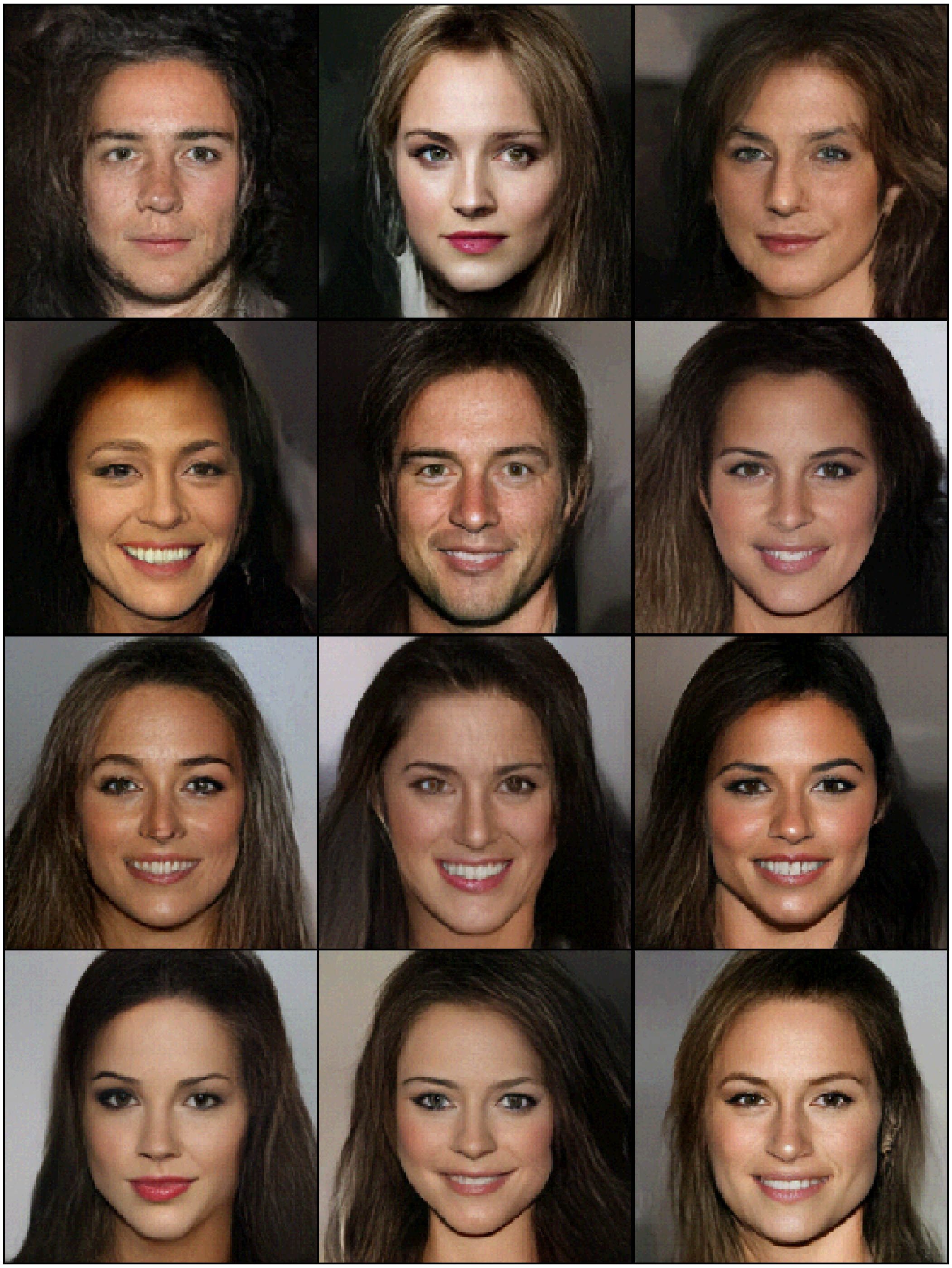


Figure 7: Synthetic celebrities sampled from 5-bit 256×256 CelebA-HQ.

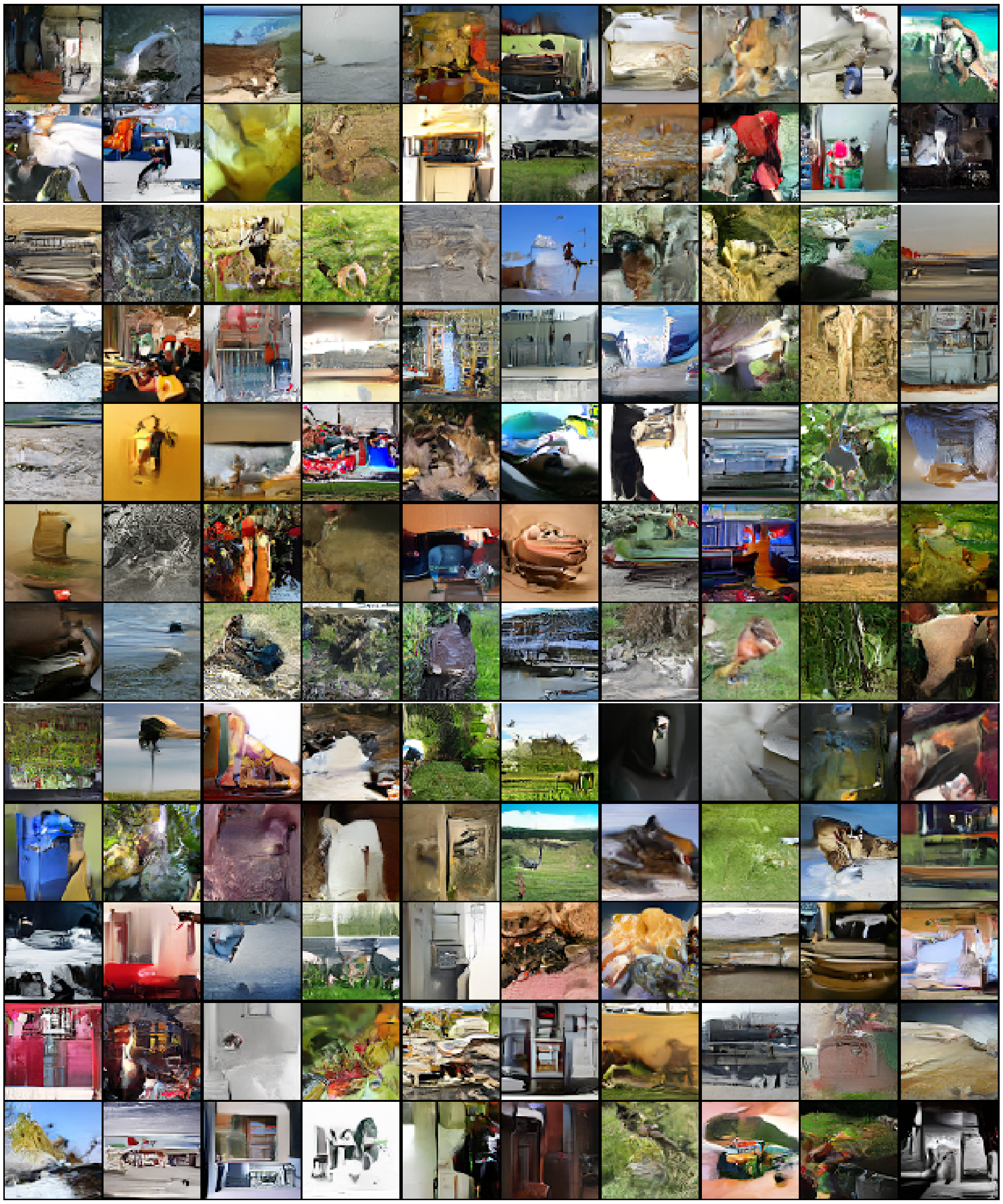


Figure 8: Samples from 8-bit imagenet 64×64 with uniform dequantization

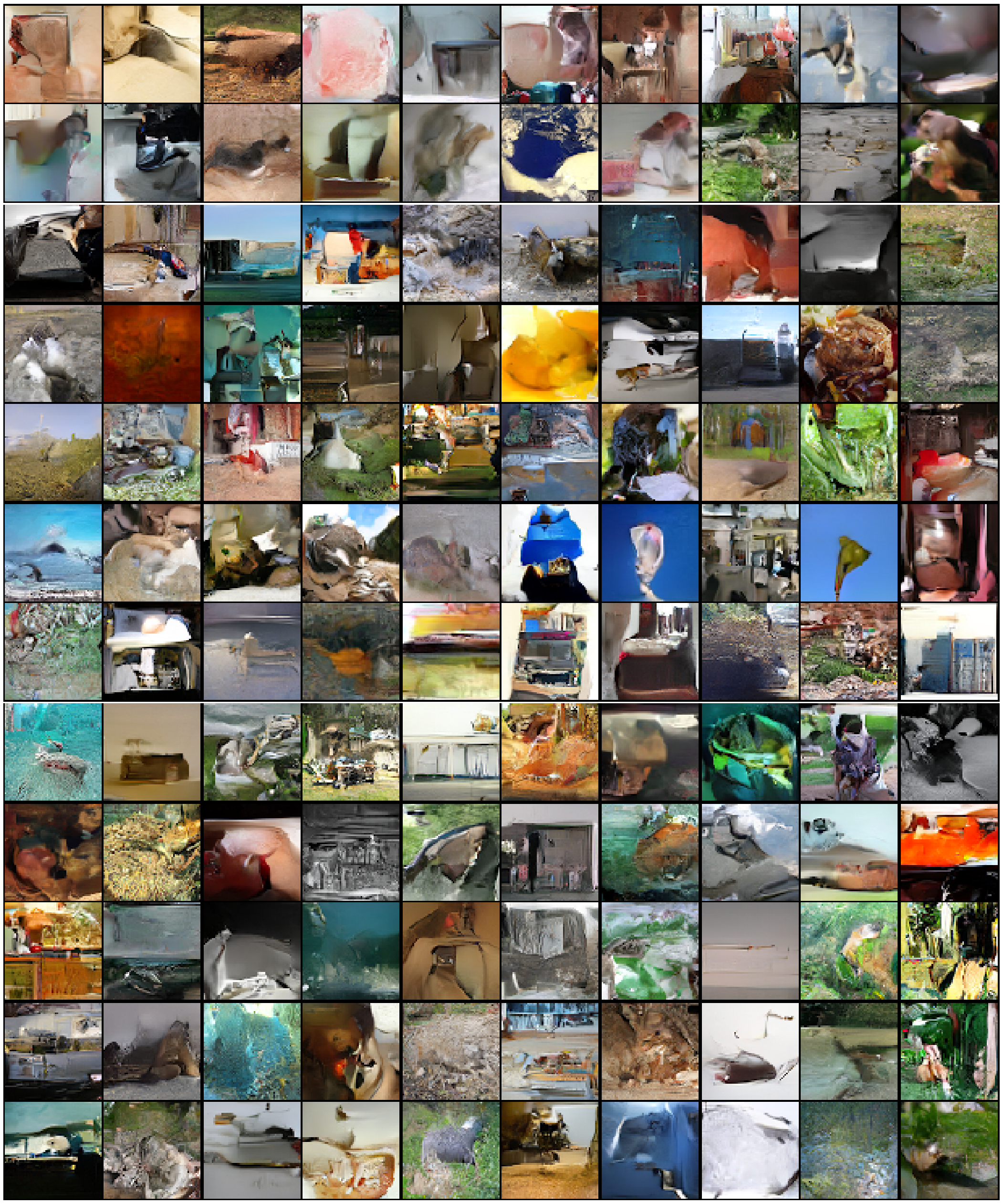


Figure 9: Samples from 8-bit imagenet 64×64 with variational dequantization