Densely connected normalizing flows

Anonymous Author(s)
Affiliation
Address
email

Abstract

Normalizing flows are bijective mappings between inputs and latent representations with a fully factorized distribution. They are very attractive due to exact likelihood evaluation and efficient sampling. However, their effective capacity is often insufficient since the bijectivity constraint limits the model width. We address this issue by incrementally padding intermediate representations with noise. We precondition the noise in accordance with previous invertible units, which we describe as cross-unit coupling. Our invertible glow-like modules express intra-unit affine coupling as a fusion of a densely connected block and Nyström self-attention. We refer to our architecture as DenseFlow since both cross-unit and intra-unit couplings rely on dense connectivity. Experiments show significant improvements due to the proposed contributions, and reveal state-of-the-art density estimation among all generative models under moderate computing budgets.

1 Introduction

One of the uttermost goals of artificial intelligence is to generate images, audio waveforms, and natural-language symbols. To achieve the desired goal, the current state of the art uses deep compositions of non-linear transformations [1, 2] known as *deep generative models* [3, 4, 5, 6, 7]. Formally, deep generative models estimate an unknown data distribution p_D given by a set of i.i.d. samples $D = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n\}$. The data distribution is approximated with a model distribution p_θ defined by the architecture of the model and a set of parameters θ. While the architecture is usually handcrafted, the set of parameters θ is obtained by optimizing the likelihood across the training distribution p_D:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x} \sim p_D}[-\ln p_\theta(\boldsymbol{x})]. \tag{1}$$

Properties of the model (e.g. efficient sampling, ability to evaluate likelihood etc.) directly depend on the definition of $p_\theta(\boldsymbol{x})$, or decision to avoid it. Early approaches consider unnormalized distribution [3] which usually requires MCMC-based sample generation [8, 9, 10] with long mixing times. Alternatively, the distribution can be autoregressively factorized [7, 11], which allows likelihood estimation and powerful but slow sample generation. VAEs [4] use a factorized variational approximation of the latent representation, which allows to learn an autoencoder by optimizing a lower bound of the likelihood. The encoder part can be either learned or held constant by hyperparameters [12, 13, 14]. In both cases the decoder part generates samples from the distribution conditioned on the latent variable. Orthogonally, GANs [5] ignore the factorization of the likelihood. Instead, the generator network learns to mimic the dataset samples by competing in a minimax game. This allows to efficiently produce high quality samples [15], which however often do not span the entire training distribution support [16]. Additionally, ignoring the factorization of p_θ implies inability to evaluate the likelihood.

Contrary to previous approaches, normalizing flows [6, 17, 18] model the likelihood using a bijective mapping to a predefined latent distribution $p(\boldsymbol{z})$, typically a multivariate Gaussian. Given the bijection

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

f_θ, the likelihood is defined using the change of variables formula:

$$p_\theta(\boldsymbol{x}) = p(\boldsymbol{z}) \left| \det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} \right|, \quad \boldsymbol{z} = f_\theta(\boldsymbol{x}). \tag{2}$$

This approach requires computation of the Jacobian determinant ($\det \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}$). Therefore, during the construction of bijective transformations, a great emphasis is placed on tractable determinant computation and efficient inverse computation [18, 19]. Due to these constraints, invertible transformations require more parameters to achieve a similar capacity compared to standard NN building blocks [20]. Still, modeling $p_\theta(\boldsymbol{x})$ using bijective formulation enables exact likelihood evaluation and efficient sample generation, which makes this approach convenient for various downstream tasks [21, 22, 23].

The bijective formulation (2) implies that the input and the latent representation have the same dimensionality. Typically, convolutional units of normalizing-flow approaches [18] internally inflate the dimensionality of the input, extract useful features, and then compress them back to the original dimensionality. Unfortunately, the capacity of such transformations is limited by input dimensionality [24]. This issue can be addressed by expressing the model as a sequence of bijective transformations [18]. However, increasing the depth alone is a suboptimal approach to improve capacity of a deep model [25]. Recent works propose to widen the flow by increasing the input dimensionality [24, 26]. We propose an effective development of that idea which further improves the performance while relaxing computational requirements.

We increase the expressiveness of normalizing flows by incremental augmentation of intermediate latent representations with Gaussian noise. The proposed cross-unit coupling applies an affine transformation to the noise, where the scaling and translation are computed from a set of previous intermediate representations. In addition, we improve intra-unit coupling by proposing a transformation which fuses the global spatial context with local correlations. The proposed image-oriented architecture improves expressiveness and computational efficiency. Our models set the new state-of-the-art result in likelihood evaluation on ImageNet32, ImageNet64 and CelebA.

2 Densely connected normalizing flows

We present a recursive view on normalizing flows and propose improvements based on incremental augmentation of latent representations, and densely connected coupling modules paired with self-attention. The improved framework is then used to develop an image-oriented architecture, which we evaluate in the experimental section.

2.1 Normalizing flows with cross-unit coupling

Normalizing flows (NF) achieve their expressiveness by stacking multiple invertible transformations [18]. We illustrate this with the scheme (3) where each two consecutive latent variables $\boldsymbol{z}_{i-1}$ and $\boldsymbol{z}_i$ are connected via a dedicated flow unit f_i. Each flow unit f_i is a bijective transformation with parameters θ_i which we omit to keep notation uncluttered. The variable $\boldsymbol{z}_0$ is typically the input $\boldsymbol{x}$ drawn from the data distribution $p_D(\boldsymbol{x})$.

$$\boldsymbol{z}_0 \xleftrightarrow{f_1} \boldsymbol{z}_1 \xleftrightarrow{f_2} \boldsymbol{z}_2 \xleftrightarrow{f_3} \cdots \xleftrightarrow{f_{i-1}} \boldsymbol{z}_i \xleftrightarrow{f_i} \cdots \xleftrightarrow{f_K} \boldsymbol{z}_K, \quad \boldsymbol{z}_K \sim \mathcal{N}(0, \mathrm{I}). \tag{3}$$

Following the change of variables formula, log likelihoods of consecutive random variables $\boldsymbol{z}_i$ and $\boldsymbol{z}_{i+1}$ can be related through the Jacobian of the corresponding transformation $\mathbf{J}_{f_{i+1}}$[18]:

$$\ln p(\boldsymbol{z}_i) = \ln p(\boldsymbol{z}_{i+1}) + \ln |\det \mathbf{J}_{f_{i+1}}|. \tag{4}$$

This relation can be seen as a recursion. The term $\ln p(\boldsymbol{z}_{i+1})$ can be recursively replaced either with another instance of (4) or evaluated under the latent distribution, which marks the termination step. This setup is characteristic for most contemporary architectures [17, 18, 19, 27].

The standard NF formulation can be expanded by augmenting the input by a noise variable $\boldsymbol{e}_i$ [24, 26]. Noise $\boldsymbol{e}_i$ subjects to some known distribution $p^*(\boldsymbol{e}_i)$, e.g. multivariate Gaussian. We further improve this approach by incrementally concatenating noise to each intermediate latent representation $\boldsymbol{z}_i$. A tractable formulation of this idea can be obtained by computing the lower bound of the likelihood $p(\boldsymbol{z}_i)$ through Monte Carlo sampling of $\boldsymbol{e}_i$:

$$\ln p(\boldsymbol{z}_i) \geq \mathbb{E}_{\boldsymbol{e}_i \sim p^*(\boldsymbol{e})} \left[\ln p(\boldsymbol{z}_i, \boldsymbol{e}_i) - \ln p^*(\boldsymbol{e}_i) \right]. \tag{5}$$

The learned distribution $p(\boldsymbol{z}_i, \boldsymbol{e}_i)$ approximates the product of the target distributions $p^*(\boldsymbol{z}_i)$ and $p^*(\boldsymbol{e}_i)$, which is explained in more detail in Appendix D. We transform the introduced noise $\boldsymbol{e}_i$ with element-wise affine transformation. Parameters of this transformation are computed by a learned non-linear transformation $g_i(\boldsymbol{z}_{<i})$ of previous representations $\boldsymbol{z}_{<i} = [\boldsymbol{z}_0, ..., \boldsymbol{z}_{i-1}]$. The resulting layer h_i can be defined as:

$$\boldsymbol{z}_i^{(\text{aug})} = h_i(\boldsymbol{z}_i, \boldsymbol{e}_i, \boldsymbol{z}_{<i}) = [\boldsymbol{z}_i, \boldsymbol{\sigma} \odot \boldsymbol{e}_i + \boldsymbol{\mu}], \quad (\boldsymbol{\mu}, \boldsymbol{\sigma}) = g_i(\boldsymbol{z}_{<i}). \tag{6}$$

Square brackets $[\cdot, \cdot]$ denote concatenation along the features dimension. In order to compute the likelihood for $(\boldsymbol{z}_i, \boldsymbol{e}_i)$, we need the determinant of the jacobian

$$\frac{\partial \boldsymbol{z}_i^{(\text{aug})}}{\partial [\boldsymbol{z}_i, \boldsymbol{e}_i]} = \begin{bmatrix} \mathrm{I} & 0 \\ 0 & \mathrm{diag}(\boldsymbol{\sigma}) \end{bmatrix}. \tag{7}$$

The resulting likelihood is

$$\ln p(\boldsymbol{z}_i, \boldsymbol{e}_i) = \ln p(\boldsymbol{z}_i^{(\text{aug})}) + \ln |\det \mathrm{diag}(\boldsymbol{\sigma})|. \tag{8}$$

We join equations (5) and (8) into a single step:

$$\ln p(\boldsymbol{z}_i) \geq \mathbb{E}_{\boldsymbol{e}_i \sim p^*(\boldsymbol{e}_i)}[\ln p(\boldsymbol{z}_i^{(\text{aug})}) - \ln p^*(\boldsymbol{e}_i) + \ln |\det \mathrm{diag}(\boldsymbol{\sigma})|]. \tag{9}$$

We refer the transformation h_i as *cross-unit coupling* since it acts as an affine coupling layer [17] over a group of previous invertible units. The latent part of the input tensor is propagated without change, while the noise part is linearly transformed. The noise transformation can be viewed as reparametrization of the distribution from which we sample the noise [4]. Obtaining the $\boldsymbol{z}_i$ from $\boldsymbol{z}_i^{(\text{aug})}$ is conveniently done by removing the noise dimensions.

Fig. 1 compares the standard normalizing flow (a) normalizing flow with input augmentation [24] (b) and the proposed incremental augmentation with cross-unit coupling (c). Each flow unit f'_i consists of several invertible modules $m_{i,j}$ and cross-unit coupling h_i. The main novelty of our architecture is that each flow unit f'_{i+1} increases the dimensionality with respect to its predecessor f'_i. Cross-unit coupling h_i augments the latent variable $\boldsymbol{z}_i$ with noise $\boldsymbol{e}_i$ which is immediately transformed with an affine transformation. Parameters of the affine transformation are obtained by an arbitrary function g_i which accepts multiple previous variables $\boldsymbol{z}_{<i}$. Note that reversing the direction does not require computing g_i since we are only interested in the value of $\boldsymbol{z}_i$. For further clarification, we show the likelihood computation for the extended framework.

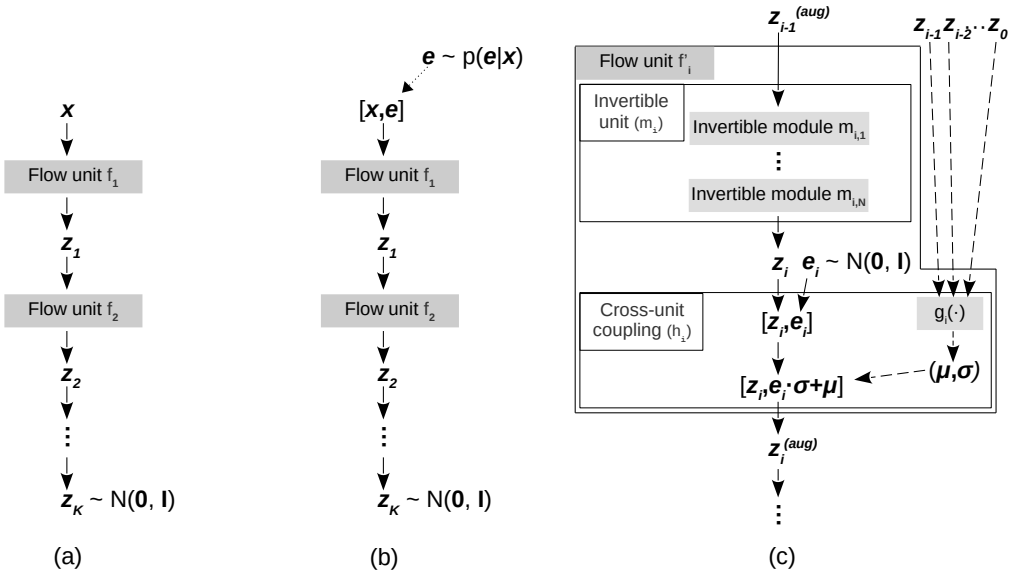

Figure 1: Standard normalizing flow [17, 18] (a), normalizing flow with augmented input [24] (b), and the proposed incremental augmentation with cross-unit coupling (c). Unlike (b) which adds noise only to the input, (c) adds noise to the output of every unit except the last.

Example 1 (Likelihood computation) *Let m_1 and m_2 be the bijective mappings from $\boldsymbol{z}_0$ to $\boldsymbol{z}_1$ and $\boldsymbol{z}_1^{(\text{aug})}$ to $\boldsymbol{z}_2$, respectively. Let h_1 be the cross-unit coupling from $\boldsymbol{z}_1$ to $\boldsymbol{z}_1^{(\text{aug})}$, $\boldsymbol{z}_1^{(\text{aug})} = [\boldsymbol{z}_1, \boldsymbol{\sigma} \odot \boldsymbol{e}_i + \boldsymbol{\mu}]$. Assume $\boldsymbol{\sigma}$ and $\boldsymbol{\mu}$ are computed by any non-invertible neural network g_i. The network accepts $\boldsymbol{z}_0$ as the input. We calculate log likelihood of the input $\boldsymbol{z}_0$ according to the following sequence of equations: [transformation, cross-unit coupling, transformation, termination].*

$$\ln p(\boldsymbol{z}_0) = \ln p(\boldsymbol{z}_1) + \ln |\det J_{f_1}|, \tag{10}$$

$$\ln p(\boldsymbol{z}_1) \geq \mathbb{E}_{\boldsymbol{e}_i \sim p^*(\boldsymbol{e}_i)}[\ln p(\boldsymbol{z}_1^{(\text{aug})}) - \ln p(\boldsymbol{e}_i) + \ln |\det \text{diag}(\boldsymbol{\sigma})|], \quad (\boldsymbol{\sigma}, \boldsymbol{\mu}) = g_1(\boldsymbol{z}_0), \tag{11}$$

$$\ln p(\boldsymbol{z}_1^{(\text{aug})}) = \ln p(\boldsymbol{z}_2) + \ln |\det J_{f_2}|, \tag{12}$$

$$\ln p(\boldsymbol{z}_2) = \ln \mathcal{N}(\boldsymbol{z}_2; 0, \text{I}). \tag{13}$$

We approximate the expectation using MC sampling with a single sample during training and a few hundreds of samples during evaluation to reduce the variance of the likelihood. Note however that our architecture generates samples with a single pass since the inverse does not require MC sampling.

We repeatedly apply the cross-unit coupling h_i throughout the architecture to achieve incremental augmentation of intermediate latent representations. Consequently, the data distribution is modeled in a latent space of higher dimensionality than the input space [24, 26]. This enables better alignment of the final latent representation with the NF prior. We materialize the proposed expansion of the normalizing flow framework by developing an image-oriented architecture which we call *DenseFlow*.

2.2 Image-oriented architecture

We propose an image-oriented architecture based on incremental augmentation with cross-unit coupling. Each DenseFlow unit f_i contains several glow-like modules $m_{i,j}$ consisting of activation normalization, 1×1 convolution and affine coupling. Different than the original glow design [19], we propose advanced transformations based on dense connectivity and fast self-attention within the coupling network. All these transformations are designed to capture complex data dependencies while keeping tractable Jacobians and efficient inverse computation. For completeness, we start by reviewing elements of the original glow module [19].

ActNorm [19] is an invertible substitute for batch normalization [30]. It performs affine transformation with per-channel scale and bias parameters:

$$\boldsymbol{y}_{i,j} = \boldsymbol{s} \odot \boldsymbol{x}_{i,j} + \boldsymbol{b}. \tag{14}$$

Scale and bias are calculated as the variance and mean of the initial minibatch.

Invertible 1×1 Convolution is a generalization of channel permutation [19]. Convolutions with 1×1 kernel are not invertible by construction. Instead, a combination of orthogonal initialization and the loss function keeps the kernel inverse numerically stable. The normalizing flow loss maximizes $\ln |\det \mathbf{J}_f|$ which is equivalent to maximizing $\sum_i \ln |\lambda_i|$, where λ_i are eigenvalues of the Jacobian. Maintaining a relatively large amplitude of the eigenvalues ensures a stable inversion. The Jacobian of this transformation can be efficiently computed by LU-decomposition [19].

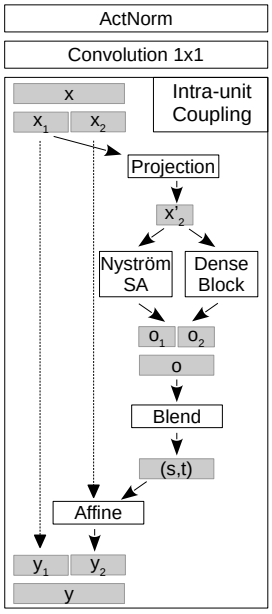

Figure 2: A glow-like module $m_{i,j}$ consist of ActNorm, 1x1 convolution and intra-unit affine coupling. The proposed intra-unit coupling fuses the global context recovered by fast self-attention [28] and local correlations extracted by densely connected convolutions [29].

Affine Coupling [18] splits the input tensor $\boldsymbol{x}$ channel-wise into two halves $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The first half is propagated without changes, while the second half is linearly transformed (15). The parameters of the linear transformation are calculated from the first half. Finally, the two results are concatenated as shown in Fig. 2.

$$\boldsymbol{y}_1 = \boldsymbol{x}_1, \quad \boldsymbol{y}_2 = \boldsymbol{s} \odot \boldsymbol{x}_2 + \boldsymbol{t}, \quad (\boldsymbol{s}, \boldsymbol{t}) = \mathit{coupling_net}(\boldsymbol{x}_1). \tag{15}$$

Parameters $\boldsymbol{s}$ and $\boldsymbol{t}$ are calculated using a trainable network which is typically implemented as a residual block [18]. However, this setting can only capture local correlations. Motivated by recent advances in discriminative architectures [29, 31, 32], we design our coupling network to fuse both global context and local correlations as shown in Fig. 2: First, we project the input into a low-dimensional manifold. Next, we feed the projected tensor to a densely-connected block [29] and self-attention module [31, 33]. The densely connected block captures the local correlations [34], while the self-attention module captures the global spatial context. Outputs of these two branches are concatenated and blended through a BN-ReLU-Conv unit. As usual, the obtained output parameterizes the affine transformation within coupling layer.

It is well known that full-fledged self-attention layers have a very large computational complexity. This is especially true in the case of normalizing flows which require many coupling layers and large latent dimensionalities. We alleviate this issue by using self-attention approximated by the Nyström method [28]. The Nyström method uses the low-rank matrix approximations instead of the actual key, query and value matrices.

Final architecture. We first join N glow-like modules ($m_i = m_{i,N} \circ \cdots \circ m_{i,1}$) containing the proposed coupling net followed by a cross-unit coupling (h_i) into an invertible DenseFlow unit (f'_i). The input to each DenseFlow unit is the output of the previous unit augmented with the noise and transformed in the cross-unit coupling fashion. The number of noise channels added is defined as the growth rate hyperparameter. Generally, the number of glow modules in latter DenseFlow units is larger compared to the initial units due to increase in latent representation size. We stack M DenseFlow units to form a DenseFlow block. The last invertible unit in the block does not have the corresponding cross-unit coupling. We stack multiple DenseFlow blocks to form a normalizing flow with a sufficient capacity. Between two consecutive blocks, we apply the space-to-channel reshaping operation and leave-out half of the dimensions [18]. We denote the developed architecture as $DenseFlow$-L-k, where L is the total number of glow-like modules in the model and k stands for the used growth rate. The developed architecture uses two independent levels of skip connections. The first level (intra-unit) is formed of skip connections inside every coupling network, while the cross-unit skips form the second level.

Fig. 3 shows the final architecture of the proposed model. Gray squares represent DenseFlow units. Cross-unit coupling is represented with blue dots and dashed skip connections. Finally, space-to-channel operations between successive DenseFlow blocks are represented by dotted squares. The proposed DenseFlow design applies invertible but less powerful transformations (e.g. convolution 1×1) on tensors of larger dimensionality. On the other hand, powerful non-invertible transformations such as coupling networks perform most of their operations on lower-dimensional tensors. This leads to a more resource-efficient architecture.

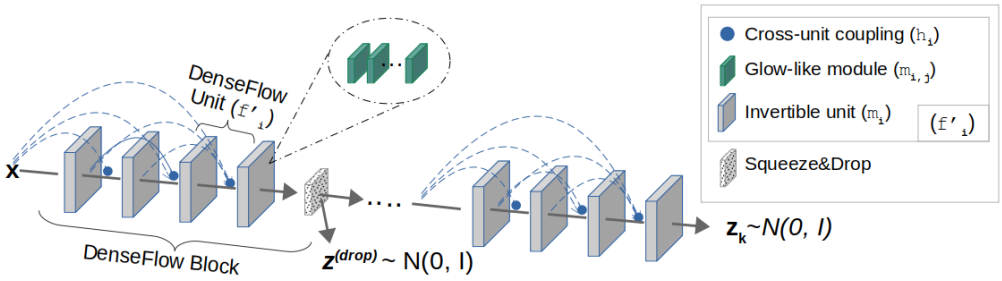

Figure 3: The proposed DenseFlow architecture. DenseFlow blocks consist of DenseFlow units (f'_i) and a Squeeze-and-Drop module [18]. DenseFlow units are densely connected through cross-unit coupling (h_i). Each DenseFlow unit includes multiple glow-like modules ($m_{i,j}$) from Fig. 2.

3 Experiments

Our experiments compare the proposed DenseFlow architecture with the state of the art. Quantitative experiments measure the accuracy of density estimation and quality of generated samples, analyze the computational complexity of model training, as well as ablate the proposed contributions. Qualitative experiments present generated samples.

3.1 Density estimation

We study the accuracy of density estimation on CIFAR-10 [35], ImageNet [36] resized to 32×32 and 64×64 pixels and CelebA [37]. Tab. 1 shows the generative performance of various contemporary models. Models are grouped into three categories based on factorization of the probability density. Among these, autoregressive models have been achieving the best performance. Image Transformer [38] has been the best on ImageNet32, while Routing transformer [39] has been the best on ImageNet64. The fourth category contains hybrid architectures which combine multiple approaches into a single model. Hybrid models have succeed to outperform many factorization-specific architectures.

The bottom row of the table presents the proposed DenseFlow architecture. We use the same DenseFlow-74-10 model in all experiments in order to illustrate the general applicability of our concepts. The first block of DenseFlow-74-10 uses 6 units with 5 glow-like modules in each DenseFlow unit, the second block uses 4 units with 6 modules, while the final block uses a single unit with 20 modules. We use a growth rate of 10 between all units. Our intra-unit couplings start with a projection to 48 channels. Subsequently, it includes a dense block with 7 densely connected layers, and the Nyström self-attention module with a single head. Since the natural images are discretized, we apply the variational dequantization procedure [27] to obtain continuous data which is suitable for normalizing flows.

Table 1: Likelihood evaluation (in bits/dim) on standard datasets.

	Method	**CIFAR-10**	**ImageNet**	**CelebA**	**ImageNet**
		32x32	32x32	64x64	64x64
Variational Autoencoders	Conv Draw [40]	3.58	4.40	-	4.10
	DVAE++ [41]	3.38	-	-	-
	IAF-VAE [42]	3.11	-	-	-
	BIVA [43]	3.08	3.96	2.48	-
	Imp. DDPM [44]	2.94	-	-	3.53
Autoregressive Models	Gated PixelCNN[45]	3.03	3.83	-	3.57
	PixelRNN [7]	3.00	3.86	-	3.63
	PixelCNN++ [11]	2.92	-	-	-
	Image Transformer [38]	2.90	3.77	2.61	-
	PixelSNAIL [46]	2.85	3.80	-	-
	SPN [47]	-	3.85	-	3.53
	Routing transformer [39]	2.95	-	-	3.43
Normalizing Flows	Real NVP [18]	3.49	4.28	3.02	3.98
	GLOW [19]	3.35	4.09	-	3.81
	Residual Flow [48]	3.28	4.01	-	3.78
	i-DenseNet [49]	3.25	3.98	-	-
	Flow++ [27]	3.08	3.86	-	3.69
	ANF [26]	3.05	3.92	-	3.66
	VFlow [24]	2.98	3.83	-	3.66
Hybrid Architectures	MaCow [50]	3.16	-	-	3.69
	SurVAE Flow [34]	3.08	4.00	-	3.70
	NVAE [51]	2.91	3.92	2.03	-
	PixelVAE++ [52]	2.90	-	-	-
	δ-VAE [53]	**2.83**	3.77	-	-
	DenseFlow-74-10 (ours)	2.98	**3.63**	**1.99**	**3.35**

On CIFAR-10, DenseFlow leverages the best results among normalizing flows corresponding to 2.98 bpd. Models trained on ImageNet32 and ImageNet64 achieve state-of-the-art results corresponding to 3.63 and 3.35 bpd respectively. The obtained results are significantly better than the previous best results (3.77 and 3.43 bpd). Finally, our model achieves the new state-of-the art result on the CelebA dataset, which corresponds to 1.99 bpd. Our model is the first model with less than 2 bpd on CelebA. The likelihood is computed using 1000 MC samples for CIFAR-10 and 200 samples for CelebA and ImageNet. The reported results are averaged over three runs with different random seeds. The variance of MC likelihood estimation is less than 0.001 for all datasets. Training details are available in Appendix C.

3.2 Computational complexity

Deep generative models require an extraordinary amount of time and computation to reach state-of-the-art results. Moreover, contemporary architectures have scaling issues. For example, VFlow [24] requires 16 GPUs and two months to be trained on the ImageNet32 dataset, while the NVAE [51] requires 24 GPUs and about 3 days. This limits downstream applications of developed models and slows down the rate of innovation in the field. In contrast, the proposed DenseFlow design places great emphasis on the scalability.

Tab. 2 compares the time and memory consumption of the proposed model with competing architectures. We compare our model with VFlow [24] and NVAE [51] due to similar generative performance on CIFAR-10 and CelebA, respectively. We note that RTX 3090 and Tesla V100 have a similar performance, while RTX2080Ti has a slightly lower performance compared to the previous two. Therefore, we can still make a fair comparison. Please note that we are unable to include approaches based on transformers [38, 39, 53] since they do not report the computational effort for model training.

Table 2: Comparison of the computational effort for model training. The proposed DenseFlow architecture requires a computational budget which is an order of magnitude smaller than for contemporary methods.

Dataset	Model	GPU type	GPUs	Duration (h)	Likelihood (bpd)
CIFAR-10	VFlow [24]	RTX 2080Ti	16	$\sim$500	2.98
	NVAE [51]	Tesla V100	8	55	2.91
	DenseFlow-74-10	RTX 3090	1	250	2.98
ImageNet32	VFlow [24]	Tesla V100	16	$\sim$1440	3.83
	NVAE [51]	Tesla V100	24	70	3.92
	DenseFlow-74-10	Tesla V100	1	310	3.63
CelebA	VFlow [24]	n/a	n/a	n/a	-
	NVAE [51]	Tesla V100	8	92	2.03
	DenseFlow-74-10	Tesla V100	1	224	1.99

3.3 Image generation

Normalizing flows can efficiently generate samples. The generation is performed in two steps. We first sample from the latent distribution and then transform the obtained latent tensor through the inverse mapping. Fig. 4 shows unconditionally generated images with the model trained on ImageNet64. Fig. 5 shows generated images using the model trained on CelebA. In this case, we modify the latent distribution by temperature scaling with factor 0.8 [38, 19, 51]. Generated images show diverse hairstyles, skin tones and backgrounds. More generated samples can be found in Appendix G. The developed DenseFlow-74-10 model generates minibatch of 128 CIFAR-10 samples for 0.96 sec. The result is averaged over 10 runs on RTX 3090.

Figure 4: Samples from DenseFlow-72-10 trained on ImageNet 64×64.

3.4 Visual quality

The ability to generate high fidelity samples is crucial for real-world applications of generative models. We measure the quality of unconditionally generated samples using the FID score [54]. The FID score requires a large corpus of generated samples in order to provide an unbiased estimate. Therefore, we leverage the number of generated samples with the number of images provided by each validation subset (10k for CIFAR-10, 20k for CelebA, and 50k for ImageNet). The samples are generated using

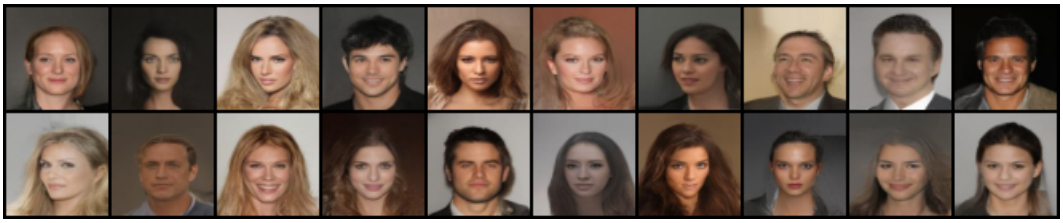

Figure 5: Samples from DenseFlow-72-10 trained on CelebA.

the model described in Sec. 3.1. The generated ImageNet32 samples achieve FID score of 39.04, the CelebA samples achieve 17.93 and CIFAR-10 samples achieve 39.27. Tab. 3 shows a comparison with FID scores of other generative models. Our model outperforms the contemporary autoregressive models [7, 55] and the majority of normalizing flows [56, 48, 19]. Additionally, the achieved FID score is comparable with the first generation of GANs. Similar to other NF models, the achieved FID score is still an order of magnitude higher than current state of the art [57]. The results for PixelCNN, DCGAN, and WGAN-GP are taken from [55].

Table 3: CIFAR-10 FID score.

	Model	**FID** $\downarrow$
Autoregressive Models	PixelCNN [7, 55]	65.93
	PixelIQN [55]	49.46
Normalizing Flows	i-ResNet [56]	65.01
	Glow [19]	46.90
	Residual flow [48]	46.37
	ANF [26]	30.60
GANs	DCGAN [15, 55]	37.11
	WGAN-GP [58, 55]	36.40
	DA-StyleGAN V2 [57]	5.79
Hybrid Architectures	SurVAE-flow [34]	49.03
	VAEBM [59]	12.19
	DenseFlow-74-10 (ours)	39.27

3.5 Ablations

Tab. 4 explores the contributions of incremental augmentation and dense connectivity in cross-unit and intra-unit coupling transforms. We decompose cross-unit coupling into incremental augmentation of the flow dimensionality (column 1) and affine noise transformation (column 2). Column 3 ablates the proposed intra-unit coupling network based on fusion of fast self-attention and a densely connected convolutional block with the original Glow coupling [19].

The bottom row of the table corresponds to a DenseFlow-45-6 model. The first DenseFlow block has 5 DenseFlow units with glow-like 3 modules per unit. The second DenseFlow block has 3 units with 5 modules, while the final block has 15 modules in a single unit. We use growth rate of 6. The top row of the table corresponds to the standard normalized flow [18, 19] with three blocks and 15 modules per block. Consequently, all models have the same number of glow-like modules. All models are trained on CIFAR-10 for 300 epochs and then fine-tuned for 10 epochs. We use the same training hyperparameters for all models. The proposed cross-unit coupling improves the density estimation from 3.42 bpd (row 1) to 3.37 bdp (row 3) for a model with the standard glow modules. When a model is equipped with our intra-unit coupling, cross-unit coupling leads to improvement from 3.14 bpd (row 4) to 3.07 bpd (row 6). Hence, the proposed cross-unit coupling improves the density estimation in all experiments. Both components of cross-unit coupling are important. Models with preconditioned noise outperform models with simple white noise (row 2 vs row 3, and row 5 vs row 6). A comparison of rows 1-3 with rows 4-6 reveals that the proposed intra-unit coupling network also yields significant improvements. We have performed two further ablation experiments with the same model. Densely connected cross-coupling contributes 0.01 bpd in comparison to preconditioning noise with respect to the previous representation only. Self-attention module contributes 0.01 bpd with respect to the model with only DenseBlock coupling on ImageNet 32×32.

Table 4: Ablations on the CIFAR-10 dataset with DenseFlow-45-6.

#	Latent variable augmentation	Pre-conditioned noise	Intra-unit coupling with two-way fusion	BPD
1	✗	✗	✗	3.42
2	✓	✗	✗	3.40
3	✓	✓	✗	3.37
4	✗	✗	✓	3.14
5	✓	✗	✓	3.08
6	✓	✓	✓	3.07

4 Related work

VFlow [24] increases the dimensionality of a normalizing flow by concatenating input with a random variable drawn from $p(\boldsymbol{e}|\boldsymbol{x})$. The resulting optimization maximizes the lower bound $\mathbb{E}_{\boldsymbol{e}\sim p(\boldsymbol{e}|\boldsymbol{x})}[\ln p(\boldsymbol{x},\boldsymbol{e}) - \ln p(\boldsymbol{e}|\boldsymbol{x})]$, where each term is implemented by a separate normalizing flow. Similarly, ANF [26] draws a connection between maximizing the joint density $p(\boldsymbol{x},\boldsymbol{e})$ and lower-bound optimization [4]. Both approaches augment only the input variable $\boldsymbol{x}$ while we augment latent representations many times throughout our models.

Surjective flows [34] decrease the computational complexity of the flow by reducing the dimensionality of deep layers. However, this also reduces the generative capacity. Our approach achieves a better generative performance under affordable computational budget due to gradual increase of the latent dimensionality and efficient coupling.

Invertible DenseNets [60, 49] apply skip connections within invertible residual blocks [56, 48]. However, this approach lacks a closed-form inverse, and therefore can generate data only through slow iterative algorithms. Our approach leverages skip connections both in cross-unit and intra-unit couplings, and supports fast analytical inverse by construction.

Models with an analytical inverse allocate most of their capacity to coupling networks [17]. Early coupling networks were implemented as residual blocks [18]. Recent work [27] increases the coupling capacity by stacking convolutional and multihead self-attention layers into a gated residual [61, 46]. However, heavy usage of self-attention radically increases the computational complexity. SurVAE [34] expresses the coupling network as a densely connected block [29] with residual connection. In comparison with [34], our intra-unit coupling omits residual connectivity, decreases the number of densely connected layers and introduces a parallel branch with Nyström self-attention. Thus, our intra-unit coupling fuses local cues with the global context.

5 Conclusion

Normalizing flows allow for principled recovery of the likelihood by evaluating factorized latent activations. However, their efficiency is hampered by the bijectivity constraint since it determines the model width. We propose to address this issue by incremental augmentation of intermediate latent representations. The introduced noise is preconditioned with respect to preceding representations throughout cross-unit affine coupling. We also propose an improved design of intra-unit coupling transformations within glow-like invertible modules. We express these transformations as a fusion of local correlations and the global context captured by self-attention. The resulting DenseFlow architecture sets the new state-of-the-art in likelihood evaluation on CelebA and ImageNet while requiring a relatively small computational budget.

6 Broader impact

This paper introduces a new generative model called DenseFlow, which can be trained to achieve state-of-the-art results under a moderate computational budget. Fast convergence and modest memory footprint lead to a relatively small environmental impact of training and favor applicability to many downstream tasks. As with most other technological advances, the results of our research could be used to pursue malicious goals. The authors disagree with any unethical applications of this work.

References

[1] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.

[4] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020.

[6] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015.

[7] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[8] Rémi Bardenet, Arnaud Doucet, and Christopher C. Holmes. On markov chain monte carlo methods for tall data. *J. Mach. Learn. Res.*, 18:47:1–47:43, 2017.

[9] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, 2002.

[10] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[11] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[12] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[14] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020.

[15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[16] Thomas Lucas, Konstantin Shmelkov, Karteek Alahari, Cordelia Schmid, and Jakob Verbeek. Adaptive density estimation for generative models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11993–12003, 2019.

[17] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[18] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[19] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10236–10245, 2018.

[20] Jörn-Henrik Jacobsen, Arnold W. M. Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[21] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark A. DePristo, Joshua V. Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14680–14691, 2019.

[22] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4723–4732. PMLR, 2019.

[23] Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Trans. Graph.*, 38(5):145:1–145:19, 2019.

[24] Jianfei Chen, Cheng Lu, Biqi Chenli, Jun Zhu, and Tian Tian. Vflow: More expressive generative flows with variational data augmentation. In *International Conference on Machine Learning*, pages 1660–1669. PMLR, 2020.

[25] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[26] Chin-Wei Huang, Laurent Dinh, and Aaron Courville. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv preprint arXiv:2002.07101*, 2020.

[27] 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. PMLR, 2019.

[28] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *CoRR*, abs/2102.03902, 2021.

[29] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[33] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[34] Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[35] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[37] 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)*, December 2015.

[38] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4052–4061. PMLR, 2018.

[39] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68, 2021.

[40] Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3549–3557, 2016.

[41] Arash Vahdat, William G. Macready, Zhengbing Bian, Amir Khoshaman, and Evgeny Andriyash. DVAE++: discrete variational autoencoders with overlapping transformations. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5042–5051. PMLR, 2018.

[42] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

[43] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. BIVA: A very deep hierarchy of latent variables for generative modeling. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6548–6558, 2019.

[44] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *CoRR*, abs/2102.09672, 2021.

[45] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4790–4798, 2016.

[46] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 863–871. PMLR, 2018.

[47] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[48] Tian Qi Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9913–9923, 2019.

[49] Yura Perugachi-Diaz, Jakub M Tomczak, and Sandjai Bhulai. Invertible densenets with concatenated lipswish. *arXiv preprint arXiv:2102.02694*, 2021.

[50] Xuezhe Ma, Xiang Kong, Shanghang Zhang, and Eduard H. Hovy. Macow: Masked convolutional generative flow. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5891–5900, 2019.

[51] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[52] Hossein Sadeghi, Evgeny Andriyash, Walter Vinci, Lorenzo Buffoni, and Mohammad H. Amin. Pixelvae++: Improved pixelvae with discrete prior. *CoRR*, abs/1908.09948, 2019.

[53] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[54] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6626–6637, 2017.

[55] Georg Ostrovski, Will Dabney, and Rémi Munos. Autoregressive quantile networks for generative modeling. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3933–3942. PMLR, 2018.

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

[57] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[58] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5767–5777, 2017.

[59] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. VAEBM: A symbiosis between variational autoencoders and energy-based models. *CoRR*, abs/2010.00654, 2020.

[60] Yura Perugachi-Diaz, Jakub M. Tomczak, and Sandjai Bhulai. Invertible densenets. In *3rd Symposium on Advances in Approximate Bayesian Inference*, pages 1–11, 2020.

[61] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[62] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and the last paragraph of Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Appendix A.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Broader impact (Section 6) and Environmental impact (Appendix B).
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 3, Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The results are averaged over multiple seeds. The variance is reported in Section 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3, Tab. 2, Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix F.
   (b) Did you mention the license of the assets? [Yes] See Appendix F.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplement.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]