Densely connected normalizing flows

Anonymous Author(s)
Affiliation
Address
email

Abstract

Normalizing flows are bijective mappings between the training data and high-dimensional latent representations with fully factorized distribution. These approaches are very attractive due to exact likelihood evaluation and fast image generation. However, their effective capacity is often insufficient since the bijectivity constraint limits the model width. We address this issue by incrementally augmenting the flow with noise. We precondition the noise in accordance with previous invertible units, which we describe as cross-unit coupling. Our invertible glow-like modules use intra-unit couplings which combine a densely connected block with 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 transformation [1, 2] also known as *deep generative models* [3, 4, 5, 6, 7].

Formally, deep generative models learn an unknown data distribution p_D corresponding to a set of i.i.d. samples $D = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$. 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 through optimization. Under the maximum likelihood setting, the optimal set of parameters θ^* is given by:

$$\theta^* = \min_{\theta \in \Theta} \mathbb{E}_{\mathbf{x} \sim p_D}[-\ln p_\theta(\mathbf{x})] \approx \min_{\theta \in \Theta} \mathbb{E}_{\mathbf{x} \in D}[-\ln p_\theta(\mathbf{x})]. \tag{1}$$

The obtained solution significantly depends on the pre-selected architecture whose design is influenced by the factorization of the probability density $p_\theta(\mathbf{x})$. Moreover, the chosen factorization, or decision to avoid it, directly defines the properties of the obtained model.

RBMs [3] model an unnormalized distribution $\hat{p}_\theta(\mathbf{x})$, which is divided by the normalization constant $Z = \int \hat{p}_\theta(\mathbf{x})\, d\mathbf{x}$ to form a valid distribution $p_\theta(\mathbf{x})$. This energy-based setup generates data points by MCMC-based sampling methods [8, 9, 10], which may have long mixing times.

Variational autoencoders [4] factorize the probability density with respect to latent variable $\mathbf{z}$: $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$. The resulting integral is intractable, so VAEs optimize a lower bound. Due to parametrization, the model has an encoder-decoder architecture. The encoder part can be learned or held constant by hyperparameters [11, 12, 13]. The decoder can only generate samples from the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$.

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

Autoregressive models [7, 14] factorize the probability density $p_\theta(\mathbf{x})$ as $p_\theta(\mathbf{x}) = \prod_i p(\mathbf{x}_i|\mathbf{x}_{<i})$, where every element $\mathbf{x}_i$ depends on all previous $\mathbf{x}_j$. This autoregressive factorization allows likelihood estimation and slow but powerful sample generation.

GANs [5] go a step aside and ignore the factorization of the probability density. Instead, the generator network learns to mimic the dataset samples by competing with the discriminator network in a minimax game defined by an appropriate loss function. Although this setting is unstable, it produces high quality samples [15]. Nevertheless, the produced samples often do not span the entire training distribution support [16]. GANs can not evaluate likelihood since they do not model the probability density.

Normalizing flows [17, 6, 18] model the probability density using bijective mapping to a predefined latent distribution $p(\mathbf{z})$, typically a multivariate Gaussian. Given the bijection f_θ, the probability density is defined using the change of variables formula:

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

This approach requires computation of Jacobian determinant ($\det \frac{\partial \mathbf{z}}{\partial \mathbf{x}}$). Therefore, during the construction of bijective transformations, a great emphasis is placed on tractable determinant computation and efficient inverse calculation [18, 19]. Due to these constraints, invertible transformations require more parameters to achieve similar capacity compared to standard NN building blocks [20].

The bijective formulation (2) implies that the input image 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 it back to the original dimensionality. Unfortunately, the capacity of such transformations is limited by the input dimensionality [21]. This issue is typically settled by stacking multiple bijective transformations into a sequence [18]. We propose to increase the expressiveness of normalizing flows by incremental augmentation of the intermediate latent representations by concatenating additional Gaussian noise dimensions. We normalize the noise with respect to the mean and variance of activations in the previous layers to form the cross-unit coupling. In addition, we propose a redesign of coupling layer's internal transformation (coupling network) which upgrades SurVAE coupling [22] to fuse both the global spatial context and the local correlations. The developed image-oriented architecture puts the new state-of-the-art results in likelihood evaluation on ImageNet32, ImageNet64 and CelebA.

2 Densely connected normalizing flow

We present a recursive view on normalizing flows and propose improvements based on incremental concatenation of latent representation 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 Expansion of normalizing flow framework

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 $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ 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 $\mathbf{z}_0$ is typically the input $\mathbf{x}$ drawn from the data distribution $p_D(\mathbf{x})$.

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

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

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

This relation can be seen as a recursion. The term $\ln p(\mathbf{z}_{i+1})$ 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 of the contemporary architectures [17, 18, 19, 21].

The standard NF formulation can be expanded by augmenting the input by a noise variable $\mathbf{e}_i$ [23, 24]. Noise $\mathbf{e}_i$ subjects to some known distribution, e.g. multivariate Gaussian. We further improve this approach by incrementally concatenating noise to each intermediate latent representation $\mathbf{z}_i$. A tractable formulation of this idea can be obtained by computing the lower bound of the likelihood $p(\mathbf{z}_i)$ through Monte Carlo sampling of $\mathbf{e}_i$ (proof in Appendix C):

$$\ln p(\mathbf{z}_i) \geq \mathbb{E}_{\mathbf{e}_i \sim p(\mathbf{e})}[\ln q([\mathbf{z}_i, \mathbf{e}_i]) - \ln p(\mathbf{e}_i)]. \tag{5}$$

We condition the introduced noise $\mathbf{e}_i$ through element-wise affine transformation. Parameters of these transformation are computed by a learned non-linear transformation of previous outputs $g_\phi(\mathbf{z}_{<i})$. The resulting invertible normalization layer h_i can be defined as follows:

$$\mathbf{z}_i^{(\mathrm{aug})} = h_i([\mathbf{z}_i, \mathbf{e}_i], \mathbf{z}_{<i}; \phi) = [\mathbf{z}_i, \boldsymbol{\sigma} \odot \mathbf{e}_i + \boldsymbol{\mu}], \quad \boldsymbol{\mu}, \boldsymbol{\sigma} = g_\phi(\mathbf{z}_{<i}) \tag{6}$$

The jacobian of the transformation h_i is simply $\frac{\partial h_i}{\partial [\mathbf{z}_i, \mathbf{e}_i]} = \begin{bmatrix} I & 0 \\ 0 & \mathrm{diag}(\boldsymbol{\sigma}) \end{bmatrix}$. Incorporating the noise transformation (6) into the change of variable formula yields:

$$\ln q([\mathbf{z}_i, \mathbf{e}_i]) = \ln q([\mathbf{z}_i, \boldsymbol{\sigma} \odot \mathbf{e}_i + \boldsymbol{\mu}]) + \ln |\det \mathrm{diag}(\boldsymbol{\sigma})| \tag{7}$$

We join equations (5) and (7) into a single recursion step defined as:

$$\ln p(\mathbf{z}_i) \geq \mathbb{E}_{\mathbf{e}_i \sim p(\mathbf{e})}[\ln q(\mathbf{z}_i^{(\mathrm{aug})}) - \ln p(\mathbf{e}_i) + \ln |\det \mathrm{diag}(\boldsymbol{\sigma})|] \tag{8}$$

We refer to this transformation as *cross-unit coupling* since it acts as a coupling layer [17] over a group of previous invertible transformations. The latent part of the input tensor is propagated without the change, while the noise part is linearly transformed. The noise transformation is also a form of a reparametrization of the distribution from which we sample the noise [4]. Obtaining the $\mathbf{z}_i$ from $\mathbf{z}_i^{(\mathrm{aug})}$ is conveniently done by removing the noise dimensions.

Figure 1 compares the standard normalizing flow (a) with input augmentation [23] (b) and the proposed incremental augmentation with cross-unit coupling (c). cross-unit coupling h_i augments latent variable $\mathbf{z}_i$ with noise $\mathbf{e}_i$ which is immediately transformed with an affine transformation. Parameters of the affine transformation are obtained by arbitrary function g_ϕ which accepts multiple previous variables $\mathbf{z}_{<i}$. Note that reversing the direction does not require computing g_ϕ since we are only interested in the value of $\mathbf{z}_i$. For further clarification, we show the likelihood computation for the described model.

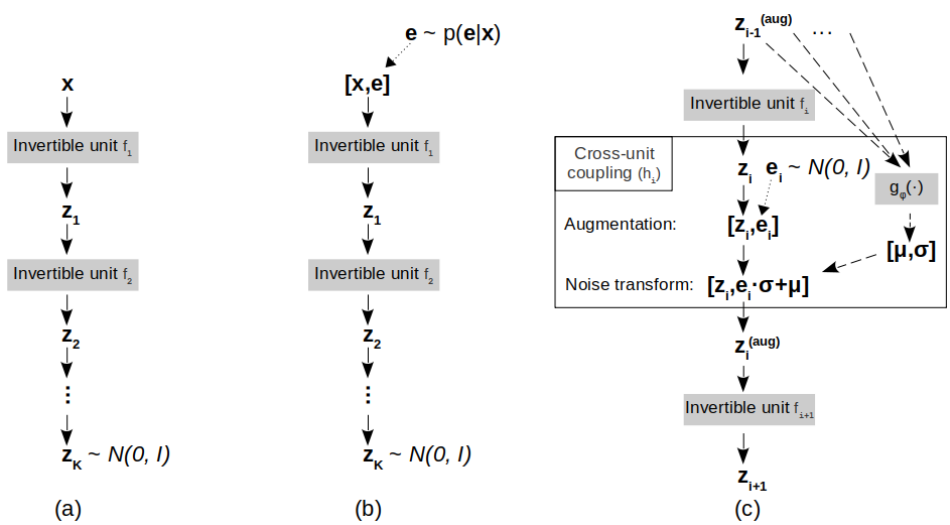

Figure 1: Standard normalizing flow [17, 18] (a), normalizing flow with augmented input [23] (b), and the proposed incremental augmentation with cross-unit coupling (c).

Example 1 (Likelihood computation) *Let f_1 and f_2 be the bijective mapping between z_0 and z_1 and $z_1^{(\mathrm{aug})}$ and z_2. Assume $\boldsymbol{\sigma}$ and $\boldsymbol{\mu}$ are computed by any non-invertible neural network g_ϕ. The*

network accepts $\boldsymbol{z}_0$ as the input. We calculate log likelihood of the input datum $\boldsymbol{z}_0$ according to the following sequence of equations: [transformation, cross-unit coupling, transformation, termination].

$$\ln p(\boldsymbol{z}_0) = \ln p(\boldsymbol{z}_1) + \ln |\det J_{f_1}| \tag{9}$$

$$\ln p(\boldsymbol{z}_1) \geq \mathbb{E}_{\boldsymbol{e}_i \sim p(\boldsymbol{e})}[\ln q(\underbrace{[\boldsymbol{z}_1, \boldsymbol{\sigma} \odot \boldsymbol{e}_i + \boldsymbol{\mu}]}_{z_1^{(\mathrm{aug})}}) - \ln p(\boldsymbol{e}_i) + \ln |\det \mathrm{diag}(\boldsymbol{\sigma})|], \quad \boldsymbol{\sigma}, \boldsymbol{\mu} = g_\phi(\boldsymbol{z}_0) \tag{10}$$

$$\ln q(\boldsymbol{z}_1^{(\mathrm{aug})}) = \ln p(\boldsymbol{z}_2) + \ln |\det J_{f_2}| \tag{11}$$

$$\ln p(\boldsymbol{z}_2) = \ln \mathcal{N}(\boldsymbol{z}_2; 0, I) \tag{12}$$

We approximate the expectation using Monte-Carlo sampling with a single sample during training and a few hundreds of samples during evaluation in order to reduce the variance of the obtained likelihood.

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 [23, 24]. This enables better alignment of the final latent representation with the normalizing flow 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 coupling units m_{ij} 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. All these transformations are designed to capture complex data dependencies while keeping tractable Jacobian and efficient inverse computation. For completeness, we start by reviewing elements of the original glow units.

ActNorm [19] is an invertible substitute for batch normalization [26]. It performs affine transformation with per-channel scale and bias parameters:

$$\mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}. \tag{13}$$

Scale and bias are calculated as the variance and mean of a given the initial minibatch. After initialization, the obtained values for scale and bias are not changed by the optimization procedure.

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 $\ln |\det J_f|$ 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].

Affine Coupling [18] splits the input tensor $\mathbf{x}$ channel-wise into two halves $\mathbf{x}_1$ and $\mathbf{x}_2$. The first half is propagated without changes while the second half is linearly transformed. The parameters of the linear transformation

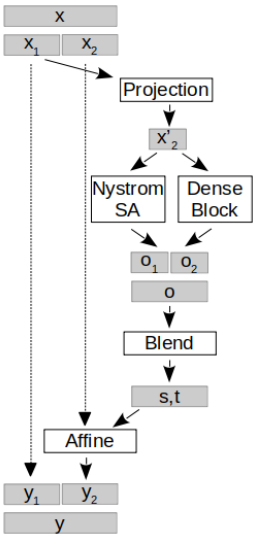

Figure 2: The proposed affine coupling fuses the global context recovered by fast self-attention [25] and local correlations extracted by densely connected convolutions [22]. This module is embedded in glow-like units m_{ij} which constitute invertible unit f_i from Fig. 1.

are calculated from the first half. Finally, the two results are concatenated as shown in Fig. 2.

$$\mathbf{y}_1 = \mathbf{x}_1, \quad \mathbf{y}_2 = \mathbf{s}(\mathbf{x}_1) \odot \mathbf{x}_2 + \mathbf{t}(\mathbf{x}_1) \tag{14}$$

The **s** and **t** functions are jointly expressed as a trainable network which is typically implemented as a residual block [18]. However, this can capture only local correlations. Motivated by recent advances in discriminative architectures [27, 28, 29], we design 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 [27, 22] and self-attention module [28] with residual connection [30]. The densely connected block captures the local correlations [22], while the self-attention module captures the global spatial context. The resulting outputs are concatenated and blended through a BN-ReLU-Conv unit. As usual, the obtained output parametrizes the affine transformation within coupling layer. In comparison with SurVAE coupling [22], we omit residual connectivity, decrease the number of dense layers and introduce self-attention.

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 single head self-attention approximated by the Nyström method [25]. The Nyström method uses the low-rank matrix approximations instead of the actual key, query and value matrices. The tensor fed to the self-attention module is not reshaped and tokenized as proposed in [30].

Figure 3 shows the final architecture of the proposed model. Gray squares represent DenseFlow units. Cross-unit coupling is designated with blue dots and dashed skip connections. Finally, space-to-channel operations between successive DenseFlow blocks is 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.

Final architecture. We first join N glow-like layers containing the proposed coupling net followed by cross-unit coupling into an invertible DenseFlow unit. 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 layers 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 unit in the block does not have the 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 factorize out half of the dimensions [18]. We denote the developed architecture as $DenseFlow\text{-}L\text{-}k$, where L is the total number of glow layers in the model and k stands for the used growth rate. The developed architecture uses two levels of skip connections. The first level is formed of skip connections inside every coupling network and the cross-unit skips form the second level of skip connections.

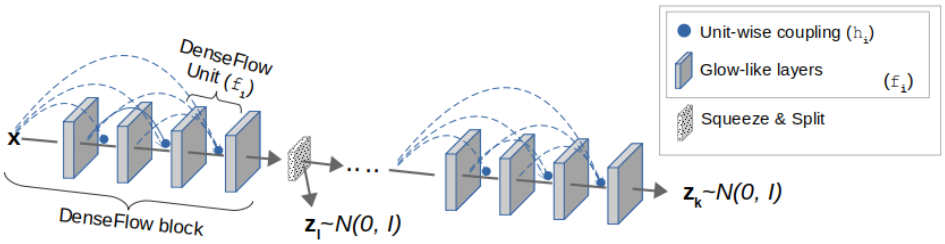

Figure 3: The proposed DenseFlow architecture. Multiple glow-like layers [19] and cross-unit coupling h_i constitute a DenseFlow unit f_i. DenseFlow units are connected with cross-unit coupling into a DenseFlow block.

3 Experiments

We demonstrate results obtained with the proposed image-oriented architecture. Quantitative results include density estimation on standard datasets, measuring the quality of generated samples, computational complexity analysis, and ablations of the proposed improvements. Qualitative results show unconditional generation of samples using trained models.

3.1 Density estimation

We study the likelihood evaluation performance of deep generative models on CIFAR10 [31], ImageNet [32] resized to 32×32 and 64×64 pixels and CelebA [33] faces. Table 1 shows the likelihood evaluation performance of various contemporary models. The models are grouped into three categories based on the factorization of the probability density. Among these, autoregressive models generally achieve the best performance. The Image Transformer [34] architecture achieves the all-time best results on ImageNet32, while the Routing transformer [35] achieves the best results on ImageNet64. The fourth category contains hybrid architectures that combine multiple approaches into a single model. Clearly, hybrid models outperform other factorization-specific architectures. The proposed DenseFlow model is based on the extended normalizing flow framework. However, since it uses MC sampling by construction, its characteristics slightly deviates from standard NF models.

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

	Method	**CIFAR-10**	**ImageNet**	**CelebA**	**ImageNet**
		32x32	32x32	64x64	64x64
Variational Autoencoders	Conv Draw [36]	3.58	4.40	-	4.10
	DVAE++ [37]	3.38	-	-	-
	IAF-VAE [38]	3.11	-	-	-
	BIVA [39]	3.08	3.96	2.48	-
	Imp. DDPM [40]	2.94	-	-	3.53
Autoregressive Models	Gated PixelCNN[41]	3.03	3.83	-	3.57
	PixelRNN [7]	3.00	3.86	-	3.63
	PixelCNN++ [14]	2.92	-	-	-
	Image Transformer [34]	2.90	3.77	2.61	-
	PixelSNAIL [42]	2.85	3.80	-	-
	SPN [43]	-	3.85	-	3.53
	Routing transformer [35]	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 [44]	3.28	4.01	-	3.78
	i-DenseNet [45]	3.25	3.98	-	-
	Flow++ [21]	3.08	3.86	-	3.69
	ANF [24]	3.05	3.92	-	3.66
	VFlow [23]	2.98	3.83	-	3.66
Hybrid Architectures	MaCow [46]	3.16	-	-	3.69
	SurVAE Flow [22]	3.08	4.00	-	3.70
	NVAE [47]	2.91	3.92	2.03	-
	PixelVAE++ [48]	2.90	-	-	-
	δ-VAE [49]	**2.83**	3.77	-	-
	DenseFlow-74-10 (ours)	2.98	**3.63**	**1.99**	**3.35**

We benchmark the proposed DenseFlow architecture performance in density estimation on all three dataset. We use the same DenseFlow-74-10 architecture for all datasets to prove general applicability. The first block uses 6 units with 5 glow layers in each unit, the second step uses 4 units with 6 glow layers, while the final block uses a single unit with 20 glow layers. We use a growth rate of 10 between all units. Within coupling blocks we project tensor to 48 channels. Dense block uses 7 dense layers, while the self-attention module has a single head. Since the natural images are discretized, we apply variational dequantization procedure [21] to obtain continuous data which is suitable for normalizing flows.

We train the proposed architecture on CIFAR10 for 580 epoch using Adamax optimizer with learning rate set to 10^{-3} and batch size 64. We apply linear warm-up of learning rate in the first 5000 epochs. During the training, learning rate is decayed for 0.9975 after every epoch. After the training, the model is fine-tuned using learning rate of $2 \cdot 10^{-5}$ for 70 epochs. Similarly, the model is trained for 20 epoch on ImageNet32, 10 epoch on ImageNet64 and 50 epochs on CelebA. For all three datasets we decay learning rate by 0.95 in every epoch. The model is fine-tuned for 2 epochs on ImageNet32, 1 epoch on ImageNet64 and 5 epochs on CelebA. We use batch size of 64 for ImageNet32 and 32 for CelebA and ImageNet64. Other hyperparameters are the same as in CIFAR10 training.

On CIFAR10, the obtained model leverages the best results among normalizing flows corresponding to 2.98 bpd. Models trained on ImageNet32 and ImageNet 64 achieve the 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 CelebA dataset, which corresponds to 1.99 bpd. Our model is the first model with less than 2 bpd on CelebA dataset. The results are obtained using 1000 MC samples for CIFAR10 and 200 samples for CelebA and ImageNet. The evaluation is additionally performed using multiple seeds. The computed variance is less than 0.001 across all datasets.

3.2 Computational complexity

Deep generative models require an extraordinary amount of time and computation to reach the state-of-the-art results. Moreover, the developed architectures have scaling issues. For example, the normalizing flow VFlow [23] requires 16 GPUs and two months to be trained on the ImageNet32 dataset, while the hybrid NVAE [47] model requires 24 GPUs and about 3 days. This limits the downstream applications of the developed models and slows down the rate of innovation in the field.

In contrast, the proposed DenseFlow design places great emphasis on the scalability of the model. Table 2 compares the time and memory consumption of the proposed model with competing architectures. We compare our model with VFlow [23] as we leverage the results on CIFAR10 and with NVAE [47] since we have similar results on CelebA. While the RTX 3090 and Tesla V100 have similar performance, the RTX2080Ti has slightly lower performance compared to the previous two. Therefore, we can still make a fair comparison of computational complexity. Note that some other interesting comparisons are not possible due to missing data.

Table 2: Computational complexity compared to contemporary methods. The proposed DenseFlow uses an order of magnitude less computation to achieve the reported results.

Dataset	**Model**	**GPU type**	**GPUs**	**Duration (h)**	**Likelihood (bpd)**
CIFAR10	VFlow [23]	RTX 2080Ti	16	∼500	2.98
	NVAE [47]	Tesla V100	8	55	2.91
	DenseFlow-74-10	RTX 3090	1	250	2.98
ImageNet32	VFlow [23]	Tesla V100	16	∼1440	3.83
	NVAE [47]	Tesla V100	24	70	3.92
	DenseFlow-74-10	Tesla V100	1	310	3.63
CelebA	VFlow [23]	n/a	n/a	n/a	-
	NVAE [47]	Tesla V100	8	92	2.03
	DenseFlow-74-10	Tesla V100	1	224	1.99

3.3 Image generation

Normalizing flows can efficiently generate images. Image generation is performed in two steps. We first sample latent distribution and then transform the obtained latent tensor into the image tensor using the inverse mapping f_θ^{-1}. Figure 4 shows unconditionally generated images with a model trained on ImageNet32 dataset. The generated images show significant diversity. Figure 5 shows generated images using the model trained on CelebA dataset. Following [34, 19, 47] we apply a temperature scaling of the latent distribution by a factor of 0.8. The generated images show diverse hairstyles, skin tones and backgrounds. More generated samples can be found in Appendix F. The developed DenseFlow-74-10 model generates minibatch of 128 CIFAR10 samples for 0.96 sec. The result is averaged over 10 runs on RTX 3090.

3.4 Samples 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 [50]. The 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 CIFAR10, 20k for CelebA, and 50k for ImageNet32). The samples are generated using the model described in Sec. 3.1. The model trained on ImageNet32 achieves FID score of 39.04, while the CelebA model achieves 17.93. Our model trained on CIFAR10 achieves 39.27. Table 3 shows the comparison of our model with other generative models. Our model outperforms the contemporary autoregressive models [7, 51] and majority of normalizing flows [52, 44, 19, 22]. Additionally, achieved FID score is comparable with the first generation of GANs. Similar to other normalizing flow models, achieved FID score is still order of magnitude higher than current state of the art [53]. Note that the results for PixelCNN, PixelIQN, DCGAN, and WGAN-GP are taken from [51].

3.5 Ablations

We perform ablation experiments to demonstrate the suitability of the proposed contributions. We compare our DenseFlow model with the standard normalizing flow framework [18, 19]. We analyze the contributions of the incremental latent variable concatenation, the affine noise transformation with corresponding skip connections and the proposed coupling network design enhancement. Alternatively to the enhanced coupling network we use the residual block [18].

Table 3: CIFAR10 FID score.

Model	**FID** $\downarrow$
PixelCNN [7, 51]	65.93
PixelIQN [51]	49.46
i-ResNet [52]	65.01
SurVAE-flow [22]	49.03
Residual flow [44]	46.37
Glow [19]	46.90
ANF [24]	30.60
DenseFlow-74-10 (ours)	39.27
DCGAN [15, 51]	37.11
WGAN-GP [54, 51]	36.40
DA-StyleGAN V2 [53]	5.79

All models are trained on CIFAR10 for 300 epochs and then fine-tuned for 10 epochs. We use the same hyperparameters and coupling network architecture for all models. All models use 3 flow steps. The DenseFlow-45-6 model is used for ablations with the following architecture. The first block has 5 units with 3 glow layers in each unit, the second block has 3 units with 5 glow layers, while the final block uses 15 glow layers and a single unit. We use a growth rate of 6 between all groups. The baseline model is developed under the standard normalizing framework and has three steps with 15 glow layers in each step. Consequently, all models have the same number of glow layers. The likelihood is evaluated using 1000 samples. Table 4 shows the obtained results. Using only the incremental concatenation of latent representation improves the likelihood estimation. Combining this operation with the affine noise transformation yields with even better results. However, the most significant improvement is obtained when the proposed coupling network design enhancement is introduced. Additional experiments with the same model on ImageNet 32×32 show that the usage of single-head Nyström self-attention block improves likelihood for 0.01 bpd compared to the pure DenseBlock based coupling.

Figure 4: Samples from DenseFlow-72-10 trained on ImageNet 32×32.

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

Table 4: Ablations on CIFAR10 dataset.

Incremental concatenation of latent representations	Normalized noise	DenseFlow coupling	BPD
✗	✗	✗	3.42
✓	✗	✗	3.40
✓	✓	✗	3.37
✗	✗	✓	3.14
✓	✗	✓	3.08
✓	✓	✓	3.07

4 Related work

Increasing the dimensionality of the latent representation is first done in [23], where the input is concatenated with a random variable drawn from $p(\mathbf{e}|\mathbf{x})$. The resulting optimization maximizes likelihood lower bound $\mathbb{E}_{\mathbf{e}\sim p(\mathbf{e}|\mathbf{x})}[\ln p([\mathbf{x}, \mathbf{e}]) - \ln p(\mathbf{e}|\mathbf{x})]$, where each term is implemented by separate normalizing flow. Similarly, [24] draws connection between maximizing likelihood of joint distribution $p([\mathbf{x}, \mathbf{e}])$ and lower bound optimization [4]. Both of this approaches augment only the input variable $\mathbf{x}$ while we apply the latent concatenation step multiple times throughout architecture. Moreover, the concatenation step can be applied at arbitrary position, hence [23, 24] can be seen as special cases of the proposed flow architecture.

Surjective transformations [22] allows the introduction of slicing, rounding and max-pooling operations into the normalizing flow framework. All of these operations reduce the dimensionality of the tensor passed to latter layers, whereas our approach gradually increase the dimensionality of the latent representation.

Invertible DenseNets [55, 45] apply skip connections within the residual block with a Lipschitz constant less than one [52, 44]. The shortcoming of this approach is the lack of the closed-form inverse, so the inverse is computed using the iterative fixed-point algorithm. Unlike i-DenseNets, our model uses the skip connections for noise centralization and has the analytical inverse by construction.

Designing a model with an analytical inverse puts most of the capacity to coupling networks [17]. Initially, the coupling network were defined as residual blocks [18]. Recently, [21] increased coupling network's capacity by stacking convolutional and multihead self-attention layers into a gated residual [56, 42]. However, heavy usage of MHA blocks radically increased both time and memory consumption. Computationally more appealing approach [22] designs coupling network using the dense block [27] with residual connections. We build upon this coupling network design to capture global dependencies and additionally reduce the memory consumption.

5 Conclusion

Normalizing flows are formed by stacking multiple invertible transformations, followed by Gaussian evaluation of the latent variable. We present an extension to this framework by incremental concatenation of intermediate latent representations using additional noise channels. The introduced noise is linearly transformed based on the statistics of previous invertible transformations. The proposed invertible transformations enhance SurVAE coupling by fusing the local correlations with the global

context captured by the introduced Nyström self-attention. We evaluate the proposed contributions in likelihood evaluation and unconditional image generation. The proposed DenseFlow architecture reveals the new state-of-the-art results in likelihood evaluation on the CelebA, ImageNet 32×32 and 64×64 while requiring less computational resources. Finally, the proposed architecture enables efficient samples generation.

6 Broader impact

This paper introduces the new generative model architecture which can be trained to achieve state-of-the-art results under moderate computational budget. The faster convergence and the smaller memory requirements reduce the environmental impact of training (See Appendix B) and widens the applicability to many downstream tasks. However, this may also potentially be used to improve many malicious applications such as deep fakes etc. Authors condemn any malicious usage 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] 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.

[12] 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.

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

[14] 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.

[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] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

[56] 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.

[57] 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.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
   (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, Table 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 E.
   (b) Did you mention the license of the assets? [Yes] See Appendix E.
   (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]

A Limitations

Augmenting the latent variable increases the dimensionality of the passed tensor in all subsequent flow steps. Aggressive application of the variable concatenation step can lead to overgrowth of the latent tensor. Therefore, the developed architecture can become computationally intractable. However, this problem can be alleviated by using an appropriate growth rate and careful application of the cross-unit coupling step.

Replacing the residual block with the proposed coupling network design increases the expressiveness of the resulting normalizing flow at cost of the increased number of parameters.

B Environmental impact

Our DenseFlow model archives competitive results with order of magnitude less computation. Still, main experiments were conducted using private infrastructure powered by public grid, which has a carbon efficiency of 0.329 kgCO_2eq/kWh. A cumulative of 1120 hours of computation was performed. According to [57], total emissions are estimated to be 110.54 kgCO_2eq.

C Proofs

Proof of equation (5)

Assume $\mathbf{e}_i$ is independent of $\mathbf{z}_i$: ($\mathbf{z}_i \perp \mathbf{e}_i$).

$$\ln p(\mathbf{z}_i) = \ln \int q([\mathbf{z}_i, \mathbf{e}_i]) d\mathbf{e}_i = \ln \int q([\mathbf{z}_i, \mathbf{e}_i]) \frac{p(\mathbf{e}_i)}{p(\mathbf{e}_i)} d\mathbf{e}_i = \ln \int p(\mathbf{e}_i) \frac{q([\mathbf{z}_i, \mathbf{e}_i])}{p(\mathbf{e}_i)} d\mathbf{e}_i \tag{15}$$

$$= \ln \mathbb{E}_{\mathbf{e}_i \sim p(\mathbf{e})} \left[\frac{q([\mathbf{z}_i, \mathbf{e}_i])}{p(\mathbf{e}_i)} \right] \tag{16}$$

$$\geq \mathbb{E}_{\mathbf{e}_i \sim p(\mathbf{e})} \left[\ln \frac{q([\mathbf{z}_i, \mathbf{e}_i])}{p(\mathbf{e}_i)} \right] = \mathbb{E}_{\mathbf{e}_i \sim p(\mathbf{e})} [\ln q([\mathbf{z}_i, \mathbf{e}_i]) - \ln p(\mathbf{e}_i)] \tag{17}$$

From (15) to (16) we apply Importance weighted sampling. From (16) to (17) we apply Jensen's inequality.

D Evidence against the training set memorization

It can be argued that increasing the dimensionality of the latent representation can lead to training set memorization. We show that this is not the case when using DenseFlow. Following [47], we first generate faces using the model trained on CelebA, and then compute the L_2 distance between the generated and the training images. The L_2 distance is computed over 42×42 center crop, so the background does not affect the similarity. Figure 6 shows the generated images in the first column, followed by the five most similar training images.

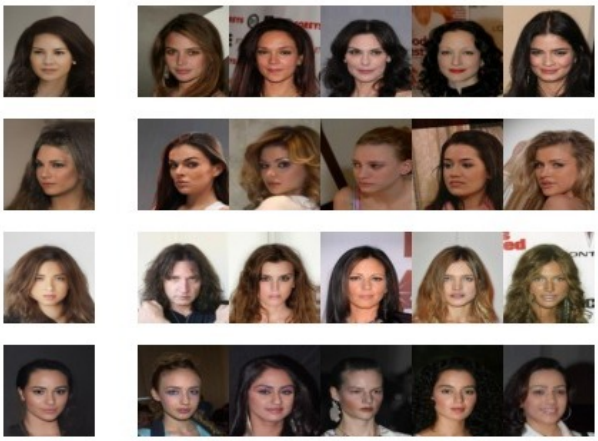

Figure 6: The generated faces and the five most similar samples from training set.

E Reproducibility & implementation

Significant part of our experimental implementation benefited from [22] whose code was publicly released under the MIT license. Our code can be found in supplementary material together with the link to model parameters.

F More samples

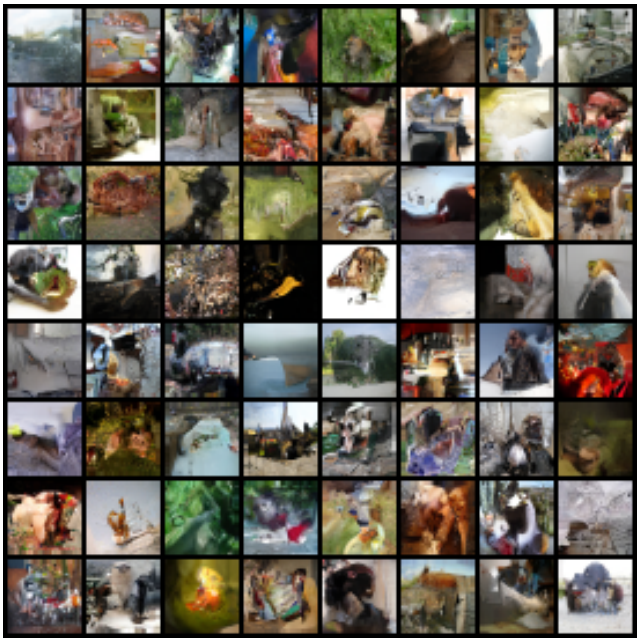

Figure 7: ImageNet 32×32 samples.

Figure 8: CIFAR10 samples.

Figure 9: CelebA samples.

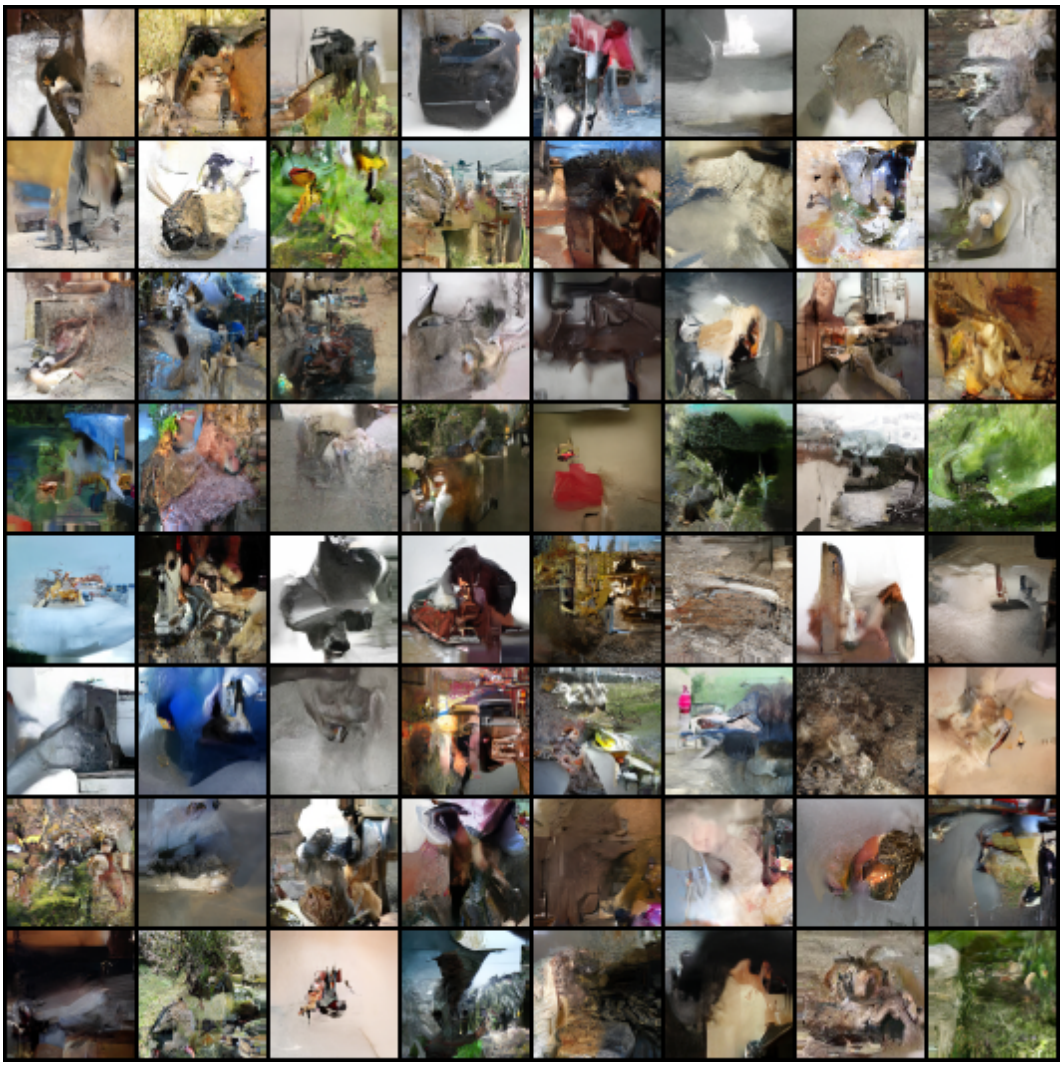

Figure 10: ImageNet 64×64 samples.