Attentive Contractive Flow with Lipschitz Constrained Self-Attention

Avideep Mukherjee[1]
avideep@cse.iitk.ac.in
Badri N. Patro[2]
badri.patro@kuleuven.be
Vinay P. Namboodiri[3]
vpn22@bath.ac.uk

[1] Indian Institute of Technology Kanpur
Kanpur, 208016
Uttar Pradesh, India

[2] KU Leuven
Oude Markt 13, 3000
Leuven, Belgium

[3] University of Bath
Claverton Down, Bath
BA2 7AY, United Kingdom

Abstract

Normalizing flows provide an elegant method for obtaining tractable density estimates from distributions using invertible transformations. The main challenge is improving the models' expressivity while keeping the invertibility constraints intact. We propose to do so via the incorporation of localized self-attention. However, conventional self-attention mechanisms do not satisfy the requirements to obtain invertible flows and cannot be naively incorporated into normalizing flows. To address this, we introduce a novel approach called Attentive Contractive Flow (ACF) which utilizes a special category of flow-based generative models - contractive flows. We demonstrate that ACF can be introduced into various state-of-the-art flow models in a plug-and-play manner. This is demonstrated to improve the representation power of these models (improving on the bits per dim metric) and result in significantly faster convergence in training them. Qualitative results, including interpolations between test images, demonstrate that samples are more realistic and capture local correlations in the data well. We evaluate the results further by performing perturbation analysis using AWGN demonstrating that ACF models (especially the dot-product variant) show better and more consistent resilience to additive noise. The code for the experiments can be found here.

1 Introduction

While deep generative models based on generative adversarial networks (GANs) and variational autoencoders (VAEs) produce state-of-the-art results showing impressive results on megapixel images, they do not have the ability to obtain exact likelihood estimates. To address this need, flow-based approaches such as real NVP [7] and invertible residual networks [1] have been proposed. However, flow-based models are still limited in their modelling capabilities as compared to GANs and VAEs. This paper focuses on improving the modelling capability of these models through the incorporation of self-attention in them. For

this purpose, a sub-category of flow-based generative models called contractive flows [32] is leveraged.

Incorporating attention in normalizing flows while maintaining the constraints of invertibility and tractable computation of the log-determinant of the Jacobian is a challenge. We propose a solution using contractive flows, called Attentive Contractive Flows (ACF), which improves modelling capability, convergence during training, and resilience to input noise.

Recent progress in generative modeling has been made through GANs such as StyleGAN2 [16] and hierarchical variational autoencoders like NVAE [39] and Flow++ [13]. While Flow++ incorporates self-attention to a slice of the image in the coupling layer, incorporating attention to the entire image space in normalizing flow-based models has not been proposed yet.

Self-attention [40] can achieve localized importance in the image and latent space, balancing the ability to model inter-dependent features with computational and statistical efficiency. During density estimation of high dimensional data, attention helps to obtain information about key positions in the image that are representative of a sample. Self-attention has been successfully incorporated in GANs[42] and VAEs[24], but incorporating it in Normalizing Flows in a generic sense is challenging due to the models' different transformation functions.

A main challenge we need to solve in order to achieve this task is to examine the invertibility of the self-attention module. In this paper, we show that while the general self-attention module [40] is not uniquely invertible, it can be made Lipschitz-continuous by replacing the dot-product operation with an L_2 norm [17] or by normalizing the whole function by a certain scalar quantity [5]. Therefore, self-attention can be made into a contraction and can be incorporated in three different contractive flows: iResNet [1], Residual Flows [3] and iDenseNets [31] each one being an improvement over the previous one. We show that the performance of all three contractive flows gets better, respectively, with Self Attention. A contractive flow uses Banach's Fixed Point Theorem to guarantee exact iterative inverses of arbitrarily complex neural networks as long as the neural network function remains a contraction. We use the variant of the self-attention mechanism inside a contractive neural network to provide a perfectly law-abiding flow-based generative model. Through this model, we are also able to obtain improved expressive capability for obtaining flows and show significantly improved performance with fewer steps as compared to other state-of-the-art NF models. Our main contributions can be summarized as follows:

- It is evident that self-attention plays a major role in the improved performance of a number of deep learning architectures. The analysis of the same for the flow-based generative approach has not been so well considered.
- In this work, we show that naive self-attention is mathematically not tractable. We show this empirically through comparisons. We need to consider the Lipschitz norm while including self-attention in normalizing flows. We demonstrate two variants of the same by incorporating either the L_2 norm or using the Lipschitz normalization and obtain attentive contractive flows.
- These have been shown to perform competitively to state-of-the-art normalizing flow methods and can be considered a complementary way for improving them.

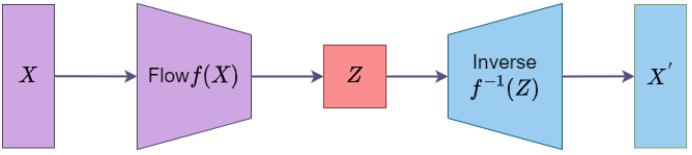

Figure 1: Attentive Contractive Flow (ACF) based generative model.

2 Preliminaries

2.1 Flow-Based Generative Models

Normalizing Flows are a class of generative models where an *'simple' parameterizable base distribution* is transformed into a *more complex approximation* for the posterior distribution [32]). This transformation is achieved by passing the base distribution through a series of invertible and bijective mappings. Let $\mathbf{z} \in \mathbb{R}^D$ and $\mathbf{y} = \mathbf{f}(\mathbf{z})$ Let $\mathbf{z}$ $q(\mathbf{z})$, be an simple base distribution. The change of variables theorem expresses a relation between the probability density functions $p_Y(\mathbf{y})$ and $q(\mathbf{z})$:

$$p_Y(\mathbf{y}) = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{y}}^{-1} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}. \tag{1}$$

If we apply a series of such mappings f_k, $k \in 1,\ldots,K$ with $K \in \mathbb{N}_+$, we obtain a **normalizing flow**. The log probability of the final distribution can thus be obtained by:

$$\log p_Y(\mathbf{y}) = \log p_Y(\mathbf{z}_K) = \log q(\mathbf{z}_0) \quad \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|. \tag{2}$$

From Equations 1 & 2, it is clearly observed that every normalizing flow architecture must satisfy two conditions. First, the transformation function should be invertible. Secondly, the log-determinant of the Jacobian should be tractable.

2.2 Contractive Flows

Residual Flows [30] are a class of invertible functions of the form:

$$\mathbf{z}' = \mathbf{f}(\mathbf{z}) = \mathbf{z} + \mathbf{g}_f(\mathbf{z}) \tag{3}$$

where $g_f : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network with parameters f. [1] proposed the transformation of Equation 3 as an invertible residual network or iResNet. Residual transformations like this can be made invertible with certain constraints on g_f. [1] show that a residual transformation is guaranteed to be invertible if the function g_f is a *contraction*. A *contraction* is a special case of a Lipschitz continuous function. A function $F : \mathbb{R}^D \to \mathbb{R}^D$ is said to be K-Lipschitz continuous when for a given distance measure d, there exists a constant K such that for two inputs $\mathbf{x_1}$ and $\mathbf{x_2}$ we have: $d(F(\mathbf{x_1}), F(\mathbf{x_2}))$ $K d(\mathbf{x_1}, \mathbf{x_2})$. The smallest such K is called the Lipschitz constant of F or Lip(F). If Lip(F) 1, then F is said to be a *contraction*. Let us consider the following equation involving the contraction in Equation 3.

$$\mathbf{F}(\hat{\mathbf{z}}) = \mathbf{z}' \quad \mathbf{g}_f(\hat{\mathbf{z}}). \tag{4}$$

Since $\mathbf{g}_f$ is contractive, $\mathbf{F}$ is also contractive with the same Lipschitz constant. Therefore, from *Banach's Fixed Point Theorem* [33], it is ensured that there exists a unique $\mathbf{z}$ such that

$\mathbf{z} = \mathbf{z}^0 \quad \mathbf{g}_f(\mathbf{z})$ which can be rearranged to $\mathbf{z}^0 = \mathbf{f}(\mathbf{z})$. Hence it follows that $\mathbf{f}$ is invertible [30]. In fact, the inversion algorithm is iteratively designed from Equation 4 as follows:

$$z_{k+1} = z^0 \quad g_f(z_k) \quad \text{for} \quad k \quad 0. \tag{5}$$

Banach's fixed-point theorem guarantees the convergence of the recursive algorithm at an exponential rate to $\mathbf{z} = \mathbf{f}^{1}(\mathbf{z}^0)$ for any arbitrary initialization of z_0 (usually it is preferred to have $z_0 = z^0$). It is clearly observed that the composition of K such residual transformations also preserves the contractive properties with the Lipschitz constant being $\tilde{O}_{i=1}^{K} L_K$, where L_K is the respective Lipschitz constant of F_K. However, there are two major challenges to building residual flows. First off, the design of the neural network function is restricted to being Lipschitz continuous, that too contraction, which limits the flexibility of the network. Secondly, the calculation of the log-determinant of the Jacobian of such a transformation cannot be efficiently computed except for automatic differentiation, which takes $O(D^3)$ time. However, the log-determinant can be approximated using the results of [12] and [41] and re-written as a power series of the trace of the Lipschitz network $\mathbf{g}_f$:

$$\log j \det J_{f_f}(\mathbf{z}) j = \log j \det \left(\mathbf{I} + J_{g_f}(\mathbf{z})\right) j + \overset{¥}{\underset{k=1}{a}} \frac{(\ 1)^{k+1}}{k} \mathrm{Tr}\left\{J_{g_f}^k(\mathbf{z})\right\} \tag{6}$$

where $J_{g_f}^k(\mathbf{z})$ is the k-th power of the Jacobian of g_f at $\mathbf{z}$.The trace can be estimated using the Hutchinson trace estimator [15]. **Residual Flows** [3] improved this method where the power series can be finitely approximated using the unbiased Russian-roulette estimator. This results in a lower requirement of computation of the power series than in the case of iResNet. Also, they introduce the LipSwish activation function to avoid derivative saturation.

Invertible DenseNets [31] use a DenseBlock as a residual layer. A DenseBlock in iDenseNet is slightly different than a standard DenseBlock and is defined as $F : \mathrm{R}^d \; 7! \; \mathrm{R}^d$ with $F(x) = x + g(x)$ where g is comprised of dense layers $fh_i g_{i=i}^{n+1}$. h_{n+1} is a 1 1 convolution to match the dimension of the output size R^d. Each h_i has two concatenated parts, the input and the transformed input:

$$h_i(x) = \begin{bmatrix} x \\ f(W_i(x)) \end{bmatrix} \tag{7}$$

where W_i is convolutional matrix and f is a non-linearity with Lip(f) 1 such as ReLU, ELU, LipSwish [3] or CLipSwish [31].

2.3 Self-Attention with Lipschitz Continuity

Lipschitz Continuity is important in neural networks to stabilize training and mitigate problems like gradient explosion. For residual flows to become invertible, they must be Lipschitz Continuous, requiring all transformation function modules to be so. Two methods of ensuring Lipschitz Continuity in self-attention modules are explored.

2.3.1 L_2 Self Attention

[17] proved that the standard self-attention module introduced by [40], or the *dot-product self-attention*, as they called it (since it involved computing a dot-product to generate the attention maps) is not Lipschitz continuous, and as a result, it is unsuitable for use in methods such as residual flows. Furthermore, they have proposed a Lipschitz Continuous variant

of the self-attention function called the L_2 Self Attention. The L_2 self-attention function on an image $X \in \mathbb{R}^{N \times D}$ (where N is the product of the height and width of the image and D is the number of channels) replaces the dot-product operation performed in dot-product self-attention module by:

$$P_{i,j} \propto \exp(L_{i,j}) = \exp\left(\frac{-\|x_i^T W^Q - x_j^T W^K\|_2^2}{\sqrt{D/H}} \right). \tag{8}$$

Here $W^Q = W^K \in \mathbb{R}^{D \times \bar{D}}$ and H is the number of heads in the multi-headed self-attention function. So, the full L_2 Multi-headed Self-Attention function is given by:

$$F(X) = [f^1(X)W^{V,1}, \quad , f^H(X)W^{V,H}]W^O \tag{9}$$

where, $f^h(X) = P^h X A_h$, $W^{V,h} \in \mathbb{R}^{D \times D/H}$, $W^O \in \mathbb{R}^{D \times D}$, P^h is defined as in Equation 8 with $W^{Q,h} = W^{K,h} \in \mathbb{R}^{D \times D/H}$ and $A_h = W^{Q,h} W^{Q,h^T} / \sqrt{D/H} \in \mathbb{R}^{D \times D}$.

2.3.2 Lipschitz Normalization

[5] introduced a normalization scheme on the dot-product self-attention itself and which makes the function Lipschitz continuous. Let $\bar{g}(X) : \mathbb{R}^{d \times n} \to \mathbb{R}^{m \times n}$ be the score function of an attention model that takes an input matrix and returns scores for each output vector $i \in 1, \quad , m$ and each input vector $j \in 1, \quad , n$. [5] normalized $\bar{g}$ by some scalar function $c : \mathbb{R}^{d \times n} \to \mathbb{R}_+$ and proved that under certain assumptions, the normalized score function $g(X) = \bar{g}(X)/c(X)$, is Lipschitz Continuous with the following bound:

$$L_F(Att) \quad e^a \sqrt{\frac{m}{n}} + a^{p-8} \tag{10}$$

where a controls the scale of all the scores.

3 Method: Contractive Flows with Self-Attention

Before defining the transformation function with self-attention, we need to make sure that the two conditions of the normalizing flow function are satisfied, i.e., the function is invertible, and the log-determinant of the Jacobian is tractable. As shown in Figure 1, on a complex image space X, and given a simple base distribution z, the model tries to learn an invertible bijective function f that maps X to Z. In our case, f is a contraction making the whole system a contractive flow. So, in order to have attention incorporated in contractive flows, the attention module needs to be inserted in between the convolutional layers in the function $\mathbf{g}_f$ of Equation 3. This would require the attention function to be a Lipschitz continuous function. Since the attention function involves computing a dot-product between the query and key matrices, it will be referred to as *dot-product self-attention*. Even if the dot-product self-attention module can be inverted, the log-determinant of the Jacobian is hard to compute. A trick to compute the non-tractable log-determinant of the Jacobian is to use the results of [41] that reduces computing the determinant to computing the trace. The result shows that for any non-singular matrix $A \in \mathbb{R}^{d \times d}$, $\ln(\det A) = \mathrm{tr}(\ln A)$ where ln is a matrix logarithm, tr is the trace of a matrix. However, this result requires the determinant be a positive quantity. It is proven that the Lipschitz-constrained perturbations of the form $x + g(x)$ yield positive Jacobian determinants [1]. Therefore, for a function $F(x) = x + g(x)$, we have, $|\det J_F(x)| = \det J_F(x)$.

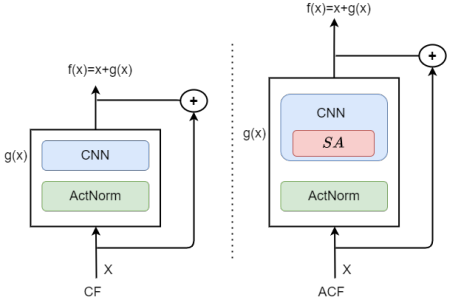

Figure 2: Model diagram comparing one step of the transformation function of a generic contractive flow (CF) (left) and a contractive flow with Self Attention (ACF)(right). SA stands for Self Attention.

Here comes the role of contractive flows, which defines transformation functions as $F(x) = x + g(x)$. But, as explained in Section 2.2, in order to make the transformation function invertible, g should not only be Lipschitz continuous but also needs to be a *contraction*, i.e., $\text{Lip}(g) < 1$. Since the dot-product self-attention is not Lipschitz continuous, as stated earlier, we use other variations of self-attention as discussed in Section 2.3.1 and 2.3.2. Equation 8 can be rewritten using matrix operations for efficient computation as:

$$P = \mathbf{S}\left(\frac{kXW^Q k_r^2 \mathbf{1}^T \quad 2XW^Q (XW^K)^T + \mathbf{1}kXW^K k_r^{2T}}{\sqrt{D/H}}\right) \tag{11}$$

where $\mathbf{S}$ is the softmax function and $jjAjj_r^2$ indicates the squared L_2 norm to each row of A, so if A 2 $\mathrm{R}^{m\ \ n}$, then $jjAjj_r^2$ 2 R^m. As discussed in Section 2.3.1 this formulation of self-attention is proven to be Lipschitz continuous and with the following bound on $\text{Lip}_2(F)$:

$$\text{Lip}_2(F) \quad \frac{p}{p}\frac{\overline{N}}{\overline{D}}(4f^{\ 1}(N \quad 1)+1)kW^Q k_2 kW^V k_2 kW^O k_2 \tag{12}$$

where $f(x) = x\exp(x+1)$ is an invertible univariate function on $x > 0$ and N is the input size. Also, $f^{\ 1}(N \quad 1) = W_0(\frac{N}{e})$ where W_0 is the Lambert W-function [17].

Hence, in order to make F a contraction, we divide F by the upper bound of $\text{Lip}_2(F)$ to obtain contractive-L_2 Self-Attention. This function satisfies every property of being a part of the transformation function of the normalizing flow. Therefore, the final attention output is given by:

$$\text{out} = g\frac{F}{\text{Lip}_2(F)} + X \tag{13}$$

where g is a learnable scalar initialized to 0. g helps the network to first attend to the local features in the neighbourhood and then gradually learn to assign more weight to the non-local evidence [42]. A detailed procedure of the forward pass of one step of the flow along with the computation of the L_2 Self Attention is provided in Algorithm 1.

Alternatively, we also applied Lipschitz Normalization on dot-product self-attention as described in Section 2.3.2. In the context of residual flows, we use the transformer variant of Lipschitz normalization. In the case of transformers, the score function, as mentioned in Section 2.3.2, is given by

$$g(X) = \frac{Q^T K}{\max fuv, uw, vwg} \tag{14}$$

where $u = jjQjj_F, v = jjK^T jj_{(¥,2)}$ and $w = jjV^T jj_{(¥,2)}$. This score function is Lipschitz with the following bound:

$$\text{Lip}_2(F) := e^{p\overline{3}}\sqrt{\frac{m}{n}} + 2^{p\overline{6}}. \tag{15}$$

Algorithm 1 Pseudo Code for a forward pass of an ACF with L_2 Self Attention [17]. SN stands for Spectral Normalization as in [1]

Require: network f, residual block g, number of power series terms n, W^Q: the query convolution, W^V,W^O : the value and out convolution respectively, H: the number of heads in the multi-headed self-attention block.

Require: X 2 $\mathbb{R}^{N\ D}$ (where N is the product of the height and width of the image and D is the number of channels)

for each residual block **do**
  Lip constraint: $\hat{W}_j := SN(W_j, X)$ for Layer W_j
  $P = \mathbf{S}\left(\frac{kXW^Q k_r^2 \mathbf{1}^T \quad 2XW_p^Q (XW^K)^T + \mathbf{1}kXW^K k_r^{2T}}{\sqrt{D/H}} \right)$
  $A := W^Q (W^Q)^T / p\overline{D}$
  $F := P \quad X \; {}_p A \quad W^L$ as in eq: 9
  $\mathrm{Lip}_2(F) := p\frac{N}{D}(4f \quad {}^1(N \quad 1)+1)jjW^Q jj_2 jjW^V jj_2 W^O jj_2$
  $\hat{W_{j+1}} := g\frac{F}{\mathrm{Lip}_2(F)} + X$: the final attention output as mentioned in eq: 13
  Draw v from $N(0, \mathbf{I})$
  $w^T = v^T$
  ln det := 0
  for k = 1 **to** n **do**
    $w^T := w^T J_g$ (vector-Jacobian product)
    ln det:= ln det $+(\quad 1)^{k+1} w^T v/k$
  end for
end for

Hence, similar to Equation 13, the score function is divided by the bound to transform the function from being Lipschitz continuous to a contraction. A detailed procedure of the forward pass of one step of the flow, along with the computation of the Lipschitz norm on dot-product self-attention, is provided in the supplementary material.

4 Experiments

We evaluate the inclusion of Self Attention in three different contractive flows: invertible ResNets, Residual Flows, and invertible DenseNets. We experiment with the combination of Lipschitz Normalization [1] on dot-product self-attention and residual flows [3] and test it on benchmark datasets. We analyze the rate of convergence of the flows both with and without attention (Fig 3), do a qualitative analysis that includes visualizations (Fig 5 and 4), and conduct ablation studies to validate the efficacy of the method (Fig 6).

4.1 Datasets

The efficacy of ACF was experimentally validated using datasets like MNIST[23], CIFAR10 [21], ImageNet32 and ImageNet64 [4], but not CelebA HQ 256 due to constrained resources. Instead, experiments were conducted on a down-sampled version: CelebA-HQ64 [25]. The standard train-test split was followed for each dataset. More detailed information about the datasets used are provided in the supplementary materials.

Model	MNIST	CIFAR10	IMAGENET32	IMAGENET64
Real NVP[8]	1.06	3.49	4.28	3.98
Glow[15]	1.05	3.35	4.09	3.81
FFJORD[10]	0.99	3.40	-	-
VFlow[8]	-	**2.98**	3.83	3.66
ANF[12]	0.93	3.05	3.92	3.66
DenseFlow[11]	-	2.98	**3.63**	**3.35**
Flow++[13]	-	3.29	3.86	3.69
iResNet[9] (iR)	1.05	3.45	-	-
iR + L_2SA (ACF)	**0.87**	3.40	-	-
ResFlow[3] (RF)	0.97	3.28	4.01	3.76
RF + L_2SA (ACF)	0.90	3.34	3.86	3.70
RF + LipNorm (ACF)	1.14	3.05	-	-
iDenseNet[21] (iD)	-	3.25	3.98	-
iD + L_2SA (ACF)	-	3.14	3.75	-

Table 1: Results [bits/dim] on standard benchmark datasets for density estimation. * are the results obtained through variational dequantization [13] which we do not compare against (following Residual Flow).

4.2 Density Estimation and Generative Modelling

We train ACF models (iResNet+L2SA, Residual Flow+L2SA, Residual Flow+ LipNorm, iDenseNet+L2SA) and their non-attentive counterparts on reported datasets. Multiple heads [8] are used in the L2 Self Attention block. The best results for MNIST and CIFAR10 datasets are achieved with H = 4 and H = 16, respectively (more in Section 4.3). ACF outperforms non-attentive contractive flows and other state-of-the-art models (Table 1), while using fewer epochs (as compared to the compared methods) and achieving faster convergence rates (Fig 3). We use the log determinant approximation (Section 2.2) in all experiments and provide further model-specific details in supplementary material.

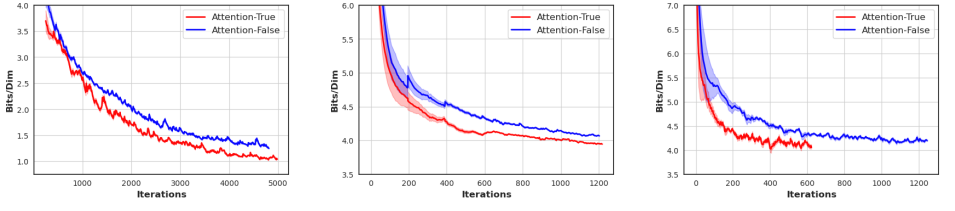

Figure 3: Convergence plots on (left) iResNet on MNIST, (middle) Residual Flows on CIFAR10 and (right) Invertible DenseNet on ImageNet32, in terms of train bits/dim across iterations. We observe that contractive flows with Self Attention converge faster (even with fewer steps of the flow) than their non-attention counterparts. All the experiments have been conducted with a random shuffling of the datasets, and the standard deviation is indicated with the shaded region after three independent trials.

4.3 Ablation Analysis: Choice of Self Attention Mechanism

The authors experimented with multi-headed L2 Self Attention in the Residual Flow architecture, comparing it against single-headed self-attention. They used Residual Flow [3] and tested the model with H = 16 and H = 4 for multi-headed L2 Self Attention [8]. They reported the bits/dim performance in Table 2 for both MNIST and CIFAR10 datasets, finding that the multi-headed self-attention block performed slightly better than its single-headed counterpart. All other hyper-parameters were kept constant for all experiments.

4.4 Qualitative Results

Figures 4 and 5 show qualitative samples and reconstructed images for ACF on MNIST, CIFAR10 and ImageNet32, demonstrating that it is capable of generating both exact re-

#heads	MNIST	CIFAR10
1	0.99	3.45
4	**0.90**	3.35
16	0.94	**3.34**

Table 2: Comparison of bits/dim results with Residual Flows + L_2 Multi-headed Self Attention varying on the number of heads tested on MNIST and CIFAR10 datasets.

constructions and realistic samples. Although there may be a discrepancy between sample quality and log-likelihood, ACF can still estimate the density of data better than other state-of-the-art generative models [37]. Additionally, ACF has a better inductive bias than autoregressive flows due to being built upon residual blocks [3]. More samples are provided in the supplementary material. We also perform interpolation between random pairs of

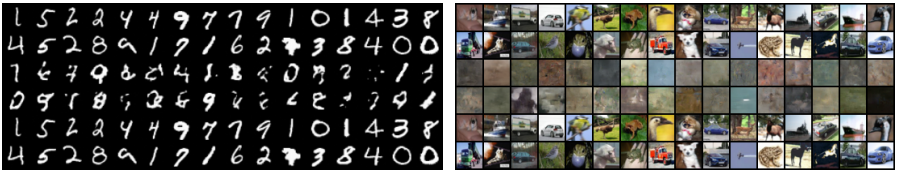

Figure 4: The images in the top two, bottom two, and middle two rows are, respectively, the real, reconstructed and generated images. (a) MNSIT results from ACF (Residual Flows + LipNorm). (b) CIFAR10 results from ACF (Residual Flows + LipNorm).

CelebA test images (say x_1 and x_2) by applying the transformation function on them to obtain corresponding samples in the latent space (say z_1 and z_2). The interpolation is performed between random pairs of images to obtain equally spaced interpolated latent samples per pair. The intermediate random variables are generated by the incremental operation: $d = z_1 + \frac{i}{N} \quad (z_2 \quad z_1) \quad 8i\ 2\ f0,1,2, \quad ,N+1g$. We follow the increment rule provided in the computation of integrated gradients [36]. The results are depicted in Figure 6. We observe that ACF is able to interpolate effectively, providing smooth transitions between pairs of diverse and unseen faces. The algorithm for interpolation and more results are provided in the supplementary material.

4.5 Classification

Following the underlying residual architecture in the attentive contractive flows, the models can be used in a discriminatory fashion and employed in tasks like image classification. ACF (iResNet+L_2SA) achieves an accuracy of 93.75% on test data of CIFAR10 in 200 fewer epochs of training compared to other flow-based methods. In Table 3, the model is compared with other contractive flows and also state-of-the-art discriminatory models that are based on residual architectures. Furthermore, since ACF preserves the contractive and residual structures as in [3] and [31], it is able to perform a discriminatory task while learning the underlying distribution of the data. Such hybrid modelling can be used in semi-supervised learning or anomaly detection. [3]

Model	Accuracy (%)
Residual Flow [3]	91.78
iDenseNet [31]	92.40
ACF (Ours) (iResNet + L_2SA)	**93.75**
ResNet v2-20 [74]	92.2
ResNet9 + Mish [74]	94.05
ResNet + ELU [33]	94.4

Table 3: Comparison of accuracy on image classification of CIFAR10 dataset.

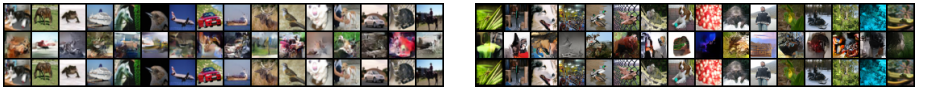

Figure 5: The images in the top, bottom, and middle row are, respectively, the real, reconstructed, and generated images. (left) CIFAR10 results from ACF (Residual Flows + L_2SA). (right) ImageNet32 results from ACF (iDenseNet + L_2SA).

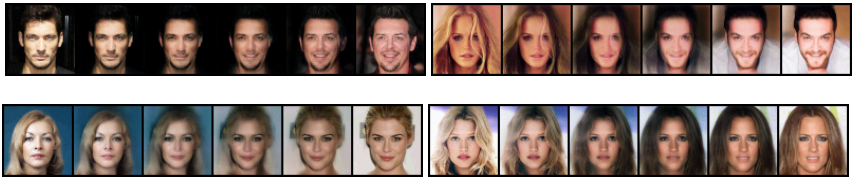

Figure 6: Interpolation between CelebA images, from one face to another, using ACF(Residual Flow + L_2SA) and the increment rule provided [36].

5 Related Works

Attention in generative models. Deep generative modelling aims to estimate the density or distribution of data to generate new samples. There are two approaches to estimation: implicit and explicit, leading to various techniques. GANs [9] are the most famous among implicit density estimation models and use an adversarial game between a generator and a discriminator to generate new samples. The Self-Attention GAN [42] uses a self-attention mechanism to improve results. Explicit density estimation models either approximate or tractably estimate the density. Apart from Restricted Boltzmann Machines [35], VAEs[18] are a well-known example of approximate density estimation, optimizing the log-likelihood of the data by maximizing the evidence lower bound. Recent work suggests incorporating self-attention on the encoder feature space of VAEs to improve the approximation of the posterior.

Normalizing Flows. Normalizing flow models estimate the density of data and are part of the change of variable models category. Unlike GANs and VAEs, normalizing flows can obtain tractable density estimates. Popular normalizing flow models include NICE [6], Real NVP [7], IAF [20], MAF [29], and Glow [19]. Recent methods like VFlow [2] or ANF[14] add additional dimensionalities to the data for better training and model expressivity. DenseFlow [11, text] proposes augmentation in the latent representation. Other approaches for generative modeling include NADE [38], MADE [22], PixelCNN, PixelRNN [28] and WaveNet [27].The Flow++ [13] model suggests implementing Self Attention in affine coupling layers of models such as real NVP [7], but only provides attention to a slice of the image or feature space. This work proposes a method to incorporate Self Attention into normalizing flows in a way that attention is applied to the whole image and feature spaces, resulting in a greater improvement in performance.

6 Conclusion and Future Direction

Our work presents a method to incorporate global self-attention into contractive flows for better modelling of complex image distributions. We show that the addition of L_2 self-attention or Lipschitz Normalization helps to attain state-of-the-art results and faster model convergence. Future research can explore how self-attention can improve specific model architectures and strengthen the representative power of normalizing flows.

References

[1] 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, 2019.

[2] 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.

[3] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pages 9916–9926, 2019.

[4] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

[5] George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In *International Conference on Machine Learning*, pages 2456–2466. PMLR, 2021.

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

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

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arxiv 2020. *arXiv preprint arXiv:2010.11929*, 2010.

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

[10] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[11] Matej Grcić, Ivan Grubišić, and Siniša Šegvić. Densely connected normalizing flows. *Advances in Neural Information Processing Systems*, 34:23968–23982, 2021.

[12] Brian Hall. *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer, 2015.

[13] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. *arXiv preprint arXiv:1902.00275*, 2019.

[14] 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.

[15] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[16] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. doi: 10.1109/cvpr42600.2020.00813. URL http://dx.doi.org/10.1109/cvpr42600.2020.00813.

[17] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. *arXiv preprint arXiv:2006.04710*, 2020.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[19] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[20] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[21] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

[22] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387, 2016.

[23] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

[24] Lizhi Lin, Xinyue Liu, and Wenxin Liang. Improving variational auto-encoder with self-attention and mutual information for image generation. In *Proceedings of the 3rd International Conference on Video and Image Processing*, ICVIP 2019, page 162–167, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450376822. doi: 10.1145/3376067.3376090. URL https://doi.org/10.1145/3376067.3376090.

[25] 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.

[26] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4:2, 2019.

[27] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[28] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

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

[30] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

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

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

[33] Walter Rudin. *Real and complex analysis*. Tata McGraw-hill education, 2006.

[34] Anish Shah, Eashan Kadam, Hena Shah, Sameer Shinde, and Sandip Shingade. Deep residual networks with exponential linear unit. In *Proceedings of the Third International Symposium on Computer Vision and the Internet*, pages 59–65, 2016.

[35] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[36] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[37] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[38] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

[39] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[41] Christopher S Withers and Saralees Nadarajah. log det a= tr log a. *International Journal of Mathematical Education in Science and Technology*, 41(8):1121–1124, 2010.

[42] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.