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

© 2022. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.
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 [3]. 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.
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 $z \in \mathbb{R}^D$ and $y = f(z)$ Let $z \sim q(z)$, be an simple base distribution. The change of variables theorem expresses a relation between the probability density functions $p_Y(y)$ and $q(z)$:

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

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:

$$\log p_Y(y) = \log p_Y(z_K) = \log q(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|. \quad (2)$$

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:

$$z' = f(z) = z + g_\phi(z) \quad (3)$$

where $g_\phi: \mathbb{R}^D \rightarrow \mathbb{R}^D$ is a neural network with parameters $\phi$. [30] 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 $g_\phi$. [30] show that a residual transformation is guaranteed to be invertible if the function $g_\phi$ is a contraction. A contraction is a special case of a Lipschitz continuous function. A function $F: \mathbb{R}^D \rightarrow \mathbb{R}^D$ is said to be $K$-Lipschitz continuous when for a given distance measure $\delta$, there exists a constant $K$ such that for two inputs $x_1$ and $x_2$ we have: $\delta(F(x_1), F(x_2)) \leq K \delta(x_1, x_2)$. The smallest such $K$ is called the Lipschitz constant of $F$ or $\text{Lip}(F)$. If $\text{Lip}(F) \leq 1$, then $F$ is said to be a contraction. Let us consider the following equation involving the contraction in Equation 3.

$$F(\hat{z}) = z' - g_\phi(\hat{z}). \quad (4)$$

Since $g_\phi$ is contractive, $F$ is also contractive with the same Lipschitz constant. Therefore, from Banach’s Fixed Point Theorem [33], it is ensured that there exists a unique $z_*$ such that
\[ z^* = z' - g_\phi(z_n) \] which can be rearranged to \( z' = f(z_n) \). Hence it follows that \( f \) is invertible \(^\square\). In fact, the inversion algorithm is iteratively designed from Equation 4 as follows:
\[
z_{k+1} = z' - g_\phi(z_k) \quad \text{for} \quad k \geq 0.
\]
Banach’s fixed-point theorem guarantees the convergence of the recursive algorithm at an exponential rate to \( z^*_k = f^{-1}(z') \) for any arbitrary initialization of \( z_0 \) (usually it is preferred to have \( z_0 = z' \)). It is clearly observed that the composition of \( K \) such residual transformations also preserves the contractive properties with the Lipschitz constant being \( \prod_{i=1}^K L_K \), 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 \( \mathcal{O}(D^3) \) 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 \( g_\phi \):
\[
\log |\det J_{f_\phi}(z)| = \log |\det (I + J_{g_\phi}(z))| + \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \text{Tr} \left\{ J^k_{g_\phi}(z) \right\}
\]
where \( J^k_{g_\phi}(z) \) is the \( k \)-th power of the Jacobian of \( g_\phi \) at \( z \). 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 \( F : \mathbb{R}^d \mapsto \mathbb{R}^d \) with \( F(x) = x + g(x) \) where \( g \) is comprised of dense layers \( \{h_i\}_{i=1}^{n+1} \). \( h_{n+1} \) is a \( 1 \times 1 \) convolution to match the dimension of the output size \( \mathbb{R}^d \). Each \( h_i \) has two concatenated parts, the input and the transformed input:
\[
h_i(x) = \begin{bmatrix} x \\
\phi(W_i(x)) \end{bmatrix}
\]
where \( W_i \) is convolutional matrix and \( \phi \) is a non-linearity with \( \text{Lip}(\phi) \leq 1 \) 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
\([4]\) 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:

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

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

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

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 $A_h = W^{Q,h} W^{Q,h\top}/\sqrt{D/H} \in \mathbb{R}^{D \times D}$.

### 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 $\tilde{g}(X) : \mathbb{R}^{d \times n} \mapsto \mathbb{R}^{m \times n}$ be the score function of an attention model that takes an input matrix and returns scores for each output vector $i \in 1, \ldots, m$ and each input vector $j \in 1, \ldots, n$. [5] normalized $\tilde{g}$ by some scalar function $c : \mathbb{R}^{d \times n} \mapsto \mathbb{R}_+$, and proved that under certain assumptions, the normalized score function $g(X) = \tilde{g}(X)/c(X)$, is Lipschitz Continuous with the following bound:

$$L_F(Att) \leq e^\alpha \sqrt{\frac{m}{n}} + \alpha \sqrt{8} \quad (10)$$

where $\alpha$ 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 $g_o$ 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 $A \in \mathbb{R}^{d \times d}$, $\ln(\det A) = \text{tr}(\ln A)$ where In is a matrix logarithm, tr is the trace of a matrix. However, this result requires the determinant to be a positive quantity. It is proven that the Lipschitz-constrained perturbations of the form $x + g(x)$ yield positive Jacobian determinants [41]. Therefore, for a function $F(x) = x + g(x)$, we have, $|\det J_F(x)| = \det J_F(x)$.
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:

$$P = S \left( - \frac{\|XWQ\|_2^2 1^T - 2XWQ(XWK)^T + 1\|XWK\|_F^2}{\sqrt{D/H}} \right)$$

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

$$\text{Lip}_2(F) \leq \sqrt{\frac{N}{\sqrt{D}}} (4\phi^{-1}(N-1) + 1)\|WQ\|_2\|WV\|_2\|WO\|_2$$

where $\phi(x) = x \exp(x+1)$ is an invertible univariate function on $x > 0$ and $N$ is the input size. Also, $\phi^{-1}(N-1) = W_0(\frac{N}{e})$ 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:

$$\text{out} = \gamma \frac{F}{\text{Lip}_2(F)} + X$$

where $\gamma$ is a learnable scalar initialized to 0. $\gamma$ 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

$$g(X) = \frac{Q^TK}{\max\{uv, uw, vw\}}$$

where $u = \|Q\|_F, v = \|K^T\|_{(\infty, 2)}$ and $w = \|V^T\|_{(\infty, 2)}$. This score function is Lipschitz with the following bound:

$$\text{Lip}_2(F) := e\sqrt{3} \sqrt{\frac{m}{n}} + 2\sqrt{6}.$$
Algorithm 1 Pseudo Code for a forward pass of an ACF with $L_2$ Self Attention \cite{equivariant}. SN stands for Spectral Normalization as in \cite{spectral}.

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

1: for each residual block do
2:  Lip constraint: $\hat{W}_j := SN(W_j, X)$ for Layer $W_j$
3:  $P = S\left(-\|XW^Q\|^2 + \|2XW^Q(XW^K)^T - 1\|XW^K\|^2/\sqrt{D/H}\right)$
4:  $A := W^Q(W^Q)^T / \sqrt{D}$
5:  $F := P \times X \times A \times W^L$ as in eq: 9
6:  $\text{Lip}_2(F) := \sqrt{\frac{N}{D}}(4\phi^{-1}(N - 1) + 1)\|W^Q\|_2\|W^V\|_2\|W^O\|_2$
7:  $\hat{W}_{j+1} := \gamma_{\text{Lip}_2(F)} + X$: the final attention output as mentioned in eq: 13
8:  Draw $v$ from $\mathcal{N}(0, I)$
9:  $w^T = v^T$
10:  $\ln \det := 0$
11:  for $k = 1$ to $n$ do
12:     $w^T := w^Tg$ (vector-Jacobian product)
13:     $\ln \det := \ln \det + (-1)^{k+1}w^Tv/k$
14:  end for
15: 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 \cite{equivariant} on dot-product self-attention and residual flows \cite{equivariant} 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\cite{mnist}, CIFAR10 \cite{cifar10}, ImageNet32 and ImageNet64 \cite{imagenet}, but not CelebA HQ 256 due to constrained resources. Instead, experiments were conducted on a down-sampled version: CelebA-HQ64 \cite{celeba}. The standard train-test split was followed for each dataset. More detailed information about the datasets used are provided in the supplementary materials.
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-
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

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+L2SA) 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]
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 [10] 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


