Supplemtary Materials: Attentive Contractive Flow

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

1 Implementation and Hyperparameter Details

The ACF models were experimented on the following datasets. Below are the details of the implementation and the hyper-parameters used. For all the experiments except ACF(iResNet) on CIFAR10, data parallelization has been performed. Tables 1, 2 and 3 provide details about the model architecture as well as the experimental environment it was trained in. The **Time** column in the tables refers to the time taken per epoch and is calculated as the product of the number of batches of the dataset and the time taken to train each batch.

Dataset	Steps	Epochs	Batch Size	Params	#GPU	GPU Type	Time/Epoch (hr)
MNIST	10	120	16	3001090	1	TITAN X - 12GB	6.2
CIFAR10	10	110	16	3001090	1	TITAN X - 12GB	8.5

Table 1: Implementation and Hyperparameter details on the respective datasets for ACF (iResNet + L_2SA)

Dataset	Steps	Epochs	Batch Size	Params	#GPU	GPU Type	Time/Epoch (hr)
MNIST	10	120	16	3597375	3	1080 Ti - 11GB	4.7
CIFAR10	10	110	16	3597375	3	TITAN X - 12GB	3.47
ImageNet32	16	1	16	5079333	2	TITAN X - 12 GB	164.6
ImageNet64	32	1	16	9031221	4	Tesla V100-16GB	266.91
5-bit CelebA - HQ	8	100	64	3103389	4	Tesla V100-16GB	0.58

Table 2: Implementation and Hyperparameter details on the respective datasets for ACF (Residual Flow + L_2SA)

Dataset	Steps	Epochs	Batch Size	Params	#GPU	GPU Type	Time/Epoch (hr)
CIFAR10	16	50	32	22577116	4	Tesla V100-16GB	9.55
ImageNet32	32	2	32	42719404	4	Tesla V100-16GB	207.59

Table 3: Implementation and Hyperparameter details on the respective datasets for ACF (iDenseNet + L_2SA)

Dataset	Steps	Epochs	Batch Size	Params	#GPU	GPU Type	Time/Epoch (hr)
MNIST	10	120	16	2730495	3	1080 Ti-11GB	2.34
CIFAR10	10	100	16	2730495	3	1080 Ti-11GB	4.89

Table 4: Implementation and Hyperparameter details on the respective datasets for ACF (Residual Flow + Lipschitz Normalization)

2 Algorithm for Contractive Flow with Lipschitz Normalized Self Attention

Algorithm 1 Pseudo Code for a forward pass of an ACF with Lipschitz Normalization [2]. 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: $X \in \mathbb{R}^{N \times D}$ (where N is the product of the height and width of the image and D is the number of channels)

for each residual block **do**
  Lip constraint: $\hat{W}_j := SN(W_j, X)$ for Layer W_j
  $\tilde{g}(X) = W^{Q^T} W^K$
  $c(X) = \max\{uv, uw, vw\}$
  $g(X) = \frac{g(X)}{c(X)}$ as in Section **??**
  $A = \text{softmax}(g(X))$
  $F = W^V A^T$
  $\text{Lip}_2(F) := e^{\sqrt{3}} \sqrt{\frac{m}{n}} + 2\sqrt{6}$
  $\hat{W_{j+1}} := \gamma \frac{F}{\text{Lip}_2(F)} + X$: the final attention output as mentioned in eq: **??**
  Draw v from $\mathcal{N}(0, \mathbf{I})$
  $w^T = v^T$
  ln det := 0
  for k = 1 **to** n **do**
    $w^T := w^T J_g$ (vector-Jacobian product)
    ln det:= ln det $+(-1)^{k+1} w^T v / k$
  end for
end for

3 Datasets

The CIFAR10 dataset consists of 60,000 32×32 colour images belonging to 10 classes, with 6000 images per class. We train the models on 50,000 images and keep the rest 10,000

images to generate the test results. The MNIST dataset contains 70,000 28×28 black and white images belonging to 10 classes, each class signifying a digit. We use 60,000 for the training of the models and the remaining 10,000 images for testing. The ImageNet32 and ImageNet64 datasets each comprise 1,281,167 training images from 1000 classes and 50,000 test images(50 images per class). The 5bit CelebA-HQ64 dataset contains 202,599 face images and is pre-processed as defined in [8].

4 Algorithm for Interpolation

Algorithm 2 Interpolation Between Two Images

Require: f: model, $nSteps$: interpolation steps)
Require: $C_1, C_2, nSteps$: number of interpolation steps
Require: **Data:** x_{C_1} and x_{C_2}
1: **for** i $\in \{0, 1, 2, \cdots, nSteps + 1\}$ **do**
2: $\delta = z_{C_1} + \frac{i}{nSteps} \times (z_{C_2} - z_{C_1})$
3: `reconstructed_image` $= f^{-1}(\delta)$
4: **end for**

5 Perturbation Analysis with Gaussian Noise

We perform ablations to study the robustness of Attentive Contractive Flows and validate the need for L_2 Self Attention over dot-product Self Attention by qualitatively and quantitatively evaluating the reconstructions of perturbed input images. For this experiment, we consider ACF(iResNet) over the MNIST dataset. The input images are perturbed by adding Gaussian noises of different variances(σ) - $0.00001, 0.0001, 0.001, 0.01, 0.1, 1$. We quantitatively evaluate the model by reporting the bits/dim values of the reconstructions for the different levels of noise. We compare the performance of L_2 Self-Attention-based ACF with dot product Self-Attention-based ACF in Table 5. The fact that dot-product self-attention is not Lipschitz-constrained leads to wrongful computation of the log-probability using the change of variable formula, which sometimes results in negative bits/dim, as we can see in Table 5.On the other hand, we observe that the model with L_2 Self Attention performs better and more consistently. The qualitative results in Figure 1 also demonstrate that the reconstructions for ACF(L_2) improve with decreasing values of σ. However, such is not the case with dot-product Self Attention. The robustness of using L_2 Self Attention with ACF is also graphically demonstrated in Figure 2(a),(b) and (c). While ACF with L_2 Self Attention follows the natural and consistent increase in bits/dim, dot product Self Attention fails to perform as expected and produces random undesirable values. We notice similar behavioural consistencies w.r.t the change in the trace values and also the log probability values when L_2 SA is used instead of dot-product SA.

Noise (σ)	ACF(dot-product SA)	ACF(L_2SA)
0.1	-726.63	42889.64
0.01	35772.76	4959.88
0.001	-9298.64	512.88
0.0001	-15126.88	47.50

Table 5: Comparison of bits/dim of reconstructions of the perturbed input images with Gaussian noises with the corresponding variances on ACF with dot-product Self Attention and ACF with L_2 Self Attention.

Figure 1: Reconstructions of perturbed input images by a Contractive Flow with Dot and L_2 Self Attention. The perturbation is done by adding Gaussian noises (ε) of different variances (σ), as shown in (b) and (c). ACF with L_2 Self Attention produces better reconstructions with a decrease in the intensity of noise. ACF with dot-product Self Attention fails to do so.

6 Qualitative Examples

Figure 3 and 4 show some more results from ACF models on various datasets. Figure 5 presents a qualitative comparison of the CelebA dataset between residual flow and ACF (Residual Flow). It can be visually observed that adding attention to the normalizing flow steps contributes in the generation of better samples. Interpolations between different CelebA face images while keeping the intermediate generations realistic are demonstrated in Figure 6.

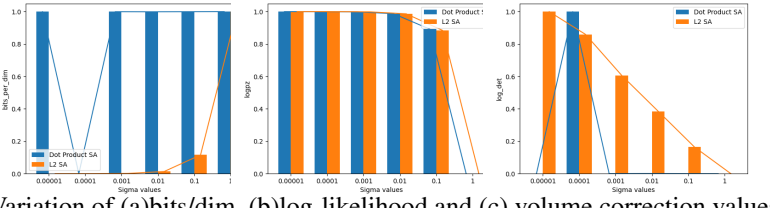
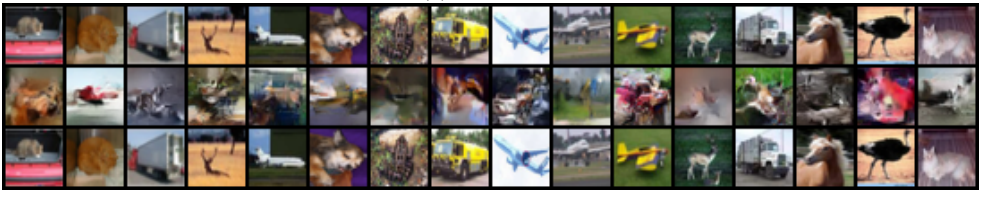


Figure 2: Variation of (a)bits/dim, (b)log-likelihood and (c) volume correction values in ACF with dot-product and ACF with L_2 Self Attention with the change in intensity of Gaussian noises applied to perturb the input MNIST images. While the variations in the values with L_2 Self Attention are always consistent, the dot-product plots again fail to produce such consistent variations.

(a) CIFAR10



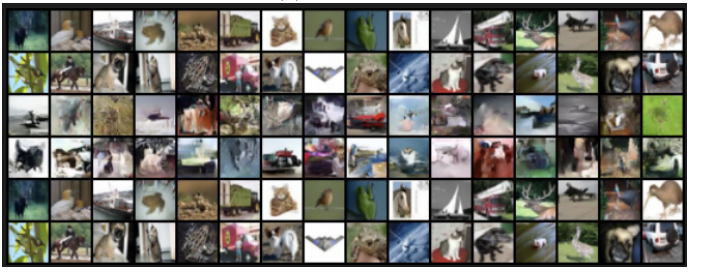
(b) ImageNet32

(c) ImageNet64

(d) 5-bit CelebA HQ

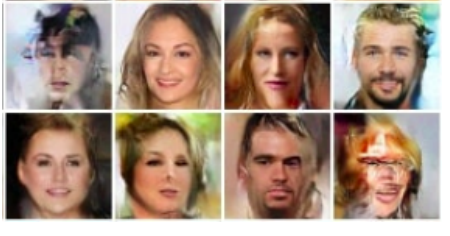
Figure 3: (a,b,c) The images in the top, bottom and middle row are respectively the real, reconstructed and generated images from ACF (Residual Flow + L_2SA). (d) The top two rows represent the read images, the bottom two rows represent their reconstruction and the middle two rows are the generated samples from ACF (Residual Flow + L_2SA).

(a) CIFAR10



(b) ImageNet32

Figure 4: Examples on CIFAR10 (above) and ImageNet32 (below) dataset from ACF(iDenseNet + L_2SA). For each dataset, the top two rows represent the read images, the bottom two rows represent their reconstruction and the middle two rows are the generated samples from the model.



(a) Samples from Residual Flow

(b) Samples from ACF

Figure 5: Comparison between CelebA samples generated from vanilla residual flow and ACF (Residual Flow + L_2SA).

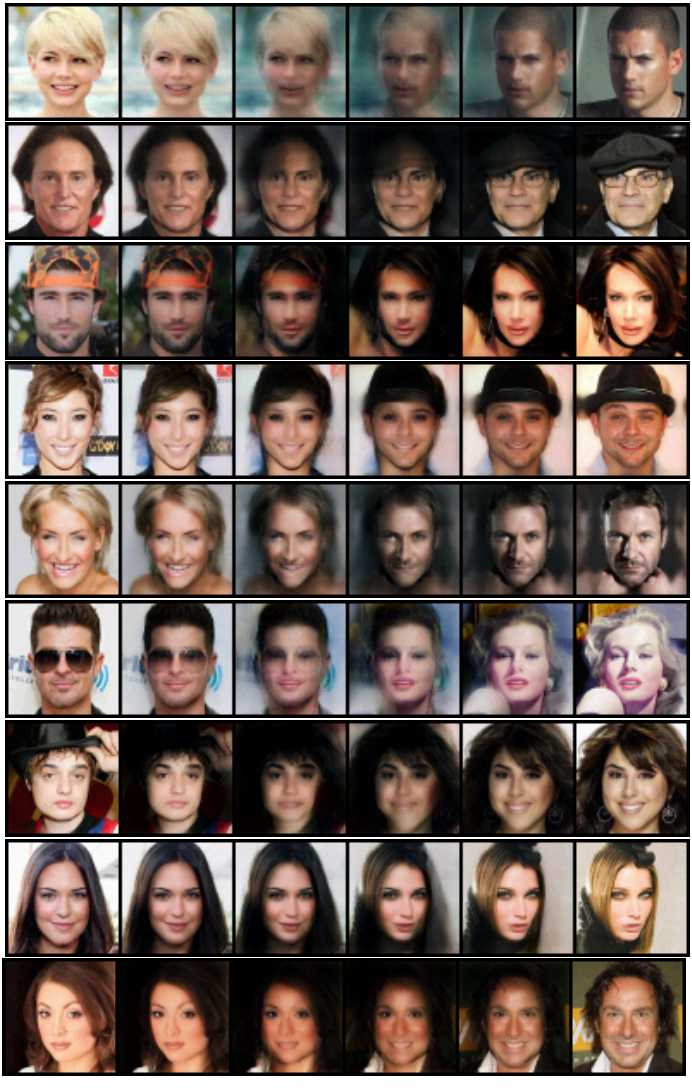


Figure 6: Interpolation between CelebA images, from one face to another using ACF(Residual Flow + L_2SA)

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] 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.

[3] 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.