LaTeX Author Guidelines for WACV Proceedings

Anonymous WACV submission

Paper ID ****

Abstract

1. Introduction

2. Related works

2.1. Normalizing flows

A normalizing flow aims at modelling an unknown data distribution, that is to be able to sample from this distribution and be able to estimate the likelyhood of an element with respect to this distribution.

To model the probability density of a random variable x a normalizing flow apply an invertible change of variable $x = g_\theta(z)$ where z is a random variable following a known distribution for instance $z \sim \mathcal{N}(0, I_d)$. Then we can get the probability of x by applying the change of variable formula

$$p_\theta(x) = p(f_\theta(x)) \left(\left| \frac{\partial f_\theta(x)}{\partial x^T} \right| \right)$$

where f_θ denotes the inverse of g_θ and $\left| \frac{\partial f_\theta(x)}{\partial x^T} \right|$ its Jacobian.

The parameters θ are learned by maximizing the true likelyhood of the dataset while the model is designed so that the function g_θ can be inverted and have its jacobian computed in a reasonnable amount of time.

2.2. Glow

RealNVP [2] defines a normalizing flow composed of a succession of invertible steps. Each of these steps can be decomposed in layers steps. Improvements for some of these layers where proposed in later articles ([5],[4]).

Actnorm The actnorm layer performs an affine transformation similar to batch normalization. It's weights are initialized so that the output of the actnorm layer have zero mean and unit variance. Then its parameters are learned without any constraint.

Permutation RealNVP proposed to use a fixed permutation to shuffle the channels as the coupling layer only act on half of the channels. Later, [5] replaced this permutation by a 1 by 1 convolution in Glow. These can easily be inverted by inverting the kernel. [4] replaced this 1x1 convolution by the so-called emerging convolution. These have the same receptive has convolution with kernel of arbitrary size. However they are computed by convolving with two successive convolutions whose kernel is masked to help the inversion operation.

Coupling layer The coupling layer is used to provide flexibility to the architecture. It's design is inspired by Feistel scheme. They are used to build an invertible layer out of any given function f. Here f is learn as a convolutional neural network.

$$y = [y_1, y_2], \quad y_1 = x1, \quad y_2 = (x_2 + f(x_1)) * \exp(g(x_1))$$

where we get x_1 and x_2 by splitting the input x along the channel axis.

3. Approach

We propose to improve normalizing flows by introducing invertible 3x3 convolutions and better coupling layers. Normalizing flows involve designing of an invertible neural network which takes a latent vector (usually sampled from a Gaussian distribution) and converts it to a realistic image by a series of invertible transformations. An advantage of such models is that one can explicitly write the probability distribution of the generated image in terms of the distribution of the latent vector. This in turn allows to design a loss function that directly optimizes the probability of generating realistic images.

3.1. Invertible Transformations

Given an invertible function $f : \mathcal{X} \to \mathcal{Y}$, and a probability distribution $\mathcal{D}$ on the input space $\mathcal{X}$, the probability distribution of the random variable $f(X)$ (where X is sam-

pled from $\mathcal{D}$) is given by:

$$p\left(f(x)\right) = p\left(x\right)\left|\frac{\partial f\left(x\right)}{\partial x^T}\right|^{-1}$$

where $\left|\frac{\partial f(x)}{\partial x^T}\right|$ its Jacobian.

3.2. Invertible 3x3 Convolution

We give necessary and sufficient conditions for an arbitrary convolution with some simple modifications on the padding, to be invertible. The inverse can also be computed by an efficient back substitution algorithm.

3.2.1 Matrix of a convolution

Definition 1 (Convolution)**.** *The convolution of an input X with shape $H \times W \times C$ with a kernel K with shape $k \times k \times C \times C$ is $Y = X * K$ of shape $H - k + 1 \times W - k + 1 \times C$ which is equal to*

$$Y_{i,j,c_o} = \sum_{l,h<k} \sum_{c_i=1}^{C} I_{i+l,j+k,c_i} K_{l,k,c_i,c_o} \tag{1}$$

.

In this setting the output Z has a smaller size than the input, to prevent this the input is padded before applying the convolution.

Definition 2 (Padding)**.** *Given an image I with shape $H \times W \times C$, the (t, b, l, r) padding of I is the image $\hat{I}$ of shape $H + t + b \times W + l + r \times C$ defined as*

$$\hat{I}_{i,j,c} = \begin{cases} I_{i-t,j-l,c} & \text{if } i - t < H \text{ and } j - l < W \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

As zero padding doesn't add any bias to the input the convolution between a padded input $\hat{I}$ and a kernel K is still a linear map between the input and the output. As such it can be described as a matrix multiplication.

An image I of shape (H, W, C) can be seen as an vector $\vec{I}$ of $\mathbb{R}^{H \times W \times C}$. In the rest of this paper we will always use the following basis $I_{i,j,c} = \vec{I}_{c+Cj+CHi}$. For any index $i \leq HWC$, let (i_y, i_x, i_c) denote the indexes that satisfy $\vec{I}_i = I_{i_y,i_x,i_c}$. Note that $i < j$ iff $(i_y, i_x, i_c) \prec (j_y, j_x, j_c)$ where $\prec$ denotes the lexicographical order over $\mathbb{R}^3$. If $C = 1$ this means that the pixel (j_y, j_x) is on the right or below the pixel (i_y, i_x).

Definition 3 (Matrix of a convolution.)**.** *Let K be a kernel of shape $k \times k \times C \times C$. The matrix of a convolution of kernel K with an input X of size $H \times W \times C$ with padding (t, b, l, r) is a matrix describing the linear map $X \mapsto \hat{X} * K$.*

3.2.2 Characterization of invertible convolutions

We consider convolution with top and left padding only. For such convolutions, we give necessary and sufficient conditions for it to be invertible.

Let K be the kernel of the convolution with shape $3 \times 3 \times N \times N$ where 3×3 is the window size and N is the number of channel. Note that number of input channels should be equal to number of output channels for it to be invertible.

We first describe our conditions for the case when $N = 1$. We prove the following theorem

Theorem 1 (Characterization for $N = 1$)**.**

$$M \textit{ is invertible iff } K_{3,3} \neq 0.$$

Proof. The proof of the theorem uses Lemma 1. Since M is lower triangular, the determinant is nothing but the product of diagonal entries which is $= K_{3,3}^{h*w}$ where h, w are the dimensions of the input/output image. □

Lemma 1. *M is a lower triangular matrix with all diagonal entries $= K_{3,3}$*

Proof. Consider any entry in the upper right half of M. That is (i, j) such that $i < j$ according to the ordering given in the definition of M. $M_{i,j}$ is nothing but the scalar weight that needs to be multiplied to the jth pixel of input which computing ith pixel of the output. The linear equation relating these two variable is as follows:

$$y_i = \sum_{l=0}^{3} \sum_{k=0}^{3} K_{3-l,3-k} x_{i_x-l,i_y-k}$$

From this equations follows that if $j_x > i_x$ or $j_y > i_y$ then the ith pixel of the output does not depend on the jth pixel of the input and thus $M_{i,j} = 0$. This also justifies that all diagonal coefficients of M are equal to $K_{3,3}$

□

We also show that Theorem 1 can be generalized to convolutions on arbitrary channels and window size. The can be found in the supplementary material.

Theorem 2 (Characterization of Invertible Convolutions)**.**

$$M \textit{ is invertible iff } det(K_{3,3}) \neq 0.$$

However when padding only on two sides the corresponding M_k is block triangular (see figure 3).

3.3. Quad-coupling

The coupling layer is used to have some flexibility as the functions used in it can be of any form. However it only combines affects half of the channels.

To overcome this issue we designed a new coupling layer inspired from generalized Feistel [3] schemes. Instead of

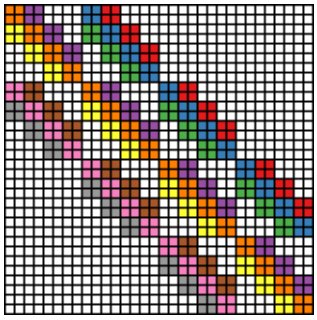

Figure 1. The matrix corresponding to a convolution with kernel of size 3 applied to an input of size 4×4 padded on both sides and with 2 channels. Zero coefficients are drawn in white, other coefficient are drawn using the same color if they are applied to the same spatial location albeit on different channels.

Figure 2. Different types of padding : on the left the standard padding (the input is in green and the white part is added zeros) that give a matrix similar to figure 1; on the left an alternative padding scheme that results in a block triangular matrix (see 3).

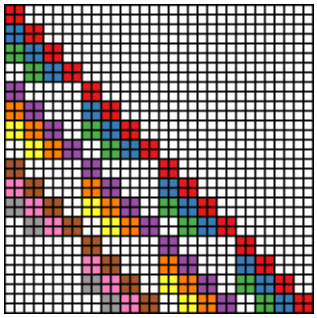

Figure 3. The matrix corresponding to a convolution with kernel of size 3 applied to an input of size 4×4 padded only on one side and with 2 channels.

dividing the input x into two blocks we divide it into four $x = [x_1, x_2, x_3, x_4]$ along the feature axis. Then we keep x_1 unchanged and use it to modify the other blocks in an autoregressive manner (see figure 5):

$$
\begin{aligned}
y_1 &= x_1 \\
y_2 &= (x_2 + f_1(x_1)) * \exp(g_1(x_1)) \\
y_3 &= (x_3 + f_2(x_1, x2)) * \exp(g_2(x_1, x_2)) \\
y_4 &= (x_4 + f_3(x_1, x_2, x_3)) * \exp(g_3(x_1, x_2, x_3))
\end{aligned}
$$

where $(f_i)_{i \leq 3}$ and $(g_i)_{i \leq 3}$ are learned. The output of the layer is obtained by concatenating the $(y_i)_{i \leq 4}$.

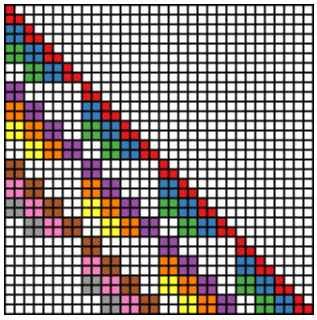

Figure 4. The matrix corresponding to a convolution with kernel of size 3 applied to an input of size 4×4 padded only on one side and with 2 channel. One of the weight of the kernel is masked.

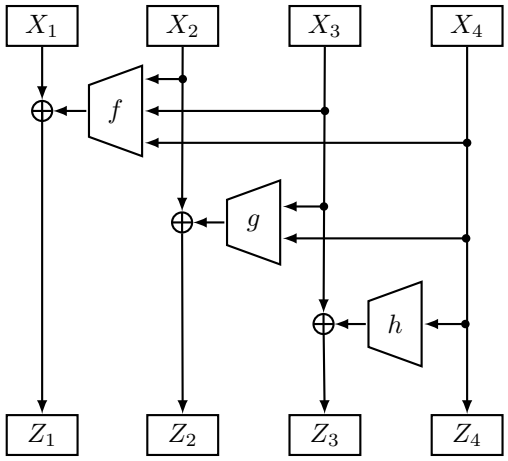

Figure 5. The quad-coupling layer, each input block X_i has the same spatial dimension as the input X but only one quarter of the channels. Each of the function f, g and h is a 3 layer convolutional network. $\bigoplus$ symbolizes a component-wise addition. The multiplicative actions are not represented here.

3.4. Image manipulation

Normalizing flows are invertible generative models. This invertibility property is really useful to get insight into the features learned by the model. We use this to modify images by acting on their latent representation.

To do this we use a dataset containing images and their attributes (for instace the Celeba dataset [7] which contains images of faces as well as attributes of these faces). Such a dataset can be modeled as two matrix : one $X \in \mathbb{R}^{N \times n}$ representing N images of size n and another $A \in \mathbb{R}^{N \times m}$ containing the N corresponding annotations corresponding to m attributes.

After training a normalizing flow using only X we can collect the output (although we could collect instead the activation of any given layer) as a matrix $Z \in \mathbb{R}^{N \times n}$. After normalizing A and Z we can compute the covariance matrix between the attributes and the latent representation of

	Emerging	3×3	Quad
Cifar10	3.3851	3.4209	3.3879
ImageNet32	4.1153	4.1085	4.0965
Galaxy	2.2722	2.2739	2.2591

Table 1. Performance achieved on the Cifar10 and Imagenet datasets after a limited number of epochs (500 for Cifar10, 600 for ImageNet32 and 1000 for Galaxy). Emerging results were obtained by using the code provided in [4], 3×3 is replacing the emerging convolutions by our " $\times$ 3 invertible convolutions and quad uses quad-coupling on top of this.

Figure 6. Example of images generated after training on the cifar dataset.

our images $C = A^T Z \in \mathbb{R}^{m \times n}$.

Then to modify an image x we compute it's latent representation z and add αC_i where C_i is the line in C corresponding to the attribute we want to modify : $C_i \in \mathbb{R}^n$ is a vector indicating in which direction a latent representation should be modified to increase the attribute in the image space.

4. Experimental results

The architecture is based on ([4]). We modified the emerging convolution layer to use our standard convolution. We also introduced the quad-coupling layer in place of the affine coupling layer. We evaluate the model on a variety of models and provide images sampled from the model.

4.1. Quantitative results

The performance of our layers was tested on CIFAR10 [6], ImageNet [8] as well as on the galaxy dataset [1].

We also tested our architecture on networks with a smaller depth ($D = 4$ or $D = 8$) which could be use when computational resources are limited as their sampling time is much lower. In this case using standard convolution and quad-coupling offers a bigger performance improvements than with bigger models (see figure 2).

4.2. Sampling times

We compared our method's sampling time against Glow and Emerging Convolutions. Our convolution still requires to solve a sequential problem to be inverted and as such need to be inverted on CPU unlike Glow that can be inverted while performing all the computation on the GPU. This explains the gap between the sampling time of our model compared to Glow. However it is still roughly two time faster than emerging convolutions; this comes from the need to solve two inversion problems in order to invert one emerging convolution layer. The quad-coupling layer doesn't affect sampling time too much.

	Glow	Emerging	3×3	Quad
Cifar10	0.58	18.4	9.3	10.8
Imagenet32	0.86	27.6	14.015	16.1

Figure 7. Time in seconds to sample 100 images. Results were obtained with Glow running on GPU and the other methods running on one CPU core.

4.3. Interpretability results

To show the interpretability of our invertible network we used the Celeba dataset [7] which provide images of faces as well as attributes corresponding to these faces. The covariance matrix between the attributes of images in the dataset and their latent representation indicates how to modify the latent representation of an image in order to add or remove features. Examples of such modifications can be seen in figures 8 and 9.

5. Conclusion

References

[1] S. Ackermann, K. Schawinski, C. Zhang, A. K. Weigel, and M. D. Turp. Using transfer learning to detect galaxy mergers. *Monthly Notices of the Royal Astronomical Society*, 479(1):415–425, 2018.

[2] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *CoRR*, abs/1605.08803, 2016.

[3] V. T. Hoang and P. Rogaway. On generalized feistel networks. In *Annual Cryptology Conference*, pages 613–630. Springer, 2010.

[4] E. Hoogeboom, R. v. d. Berg, and M. Welling. Emerging convolutions for generative normalizing flows. *arXiv preprint arXiv:1901.11137*, 2019.

[5] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[6] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[7] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Dataset	Emerging		Ours		Depth
	Performance	Sampling time	Performance	Sampling time	
Cifar10	3.52	2.45	3.49	1.31	4
Imagenet32	4.30		4.25		
Cifar10	3.47	4.94	3.46	2.76	8
Imagenet32	4.20		4.18		

Table 2. Performance with smaller networks. The performance is expressed in bits per dimension and the sampling time is the time in seconds needed to sample 100 images. All networks were trained for 600 epochs.

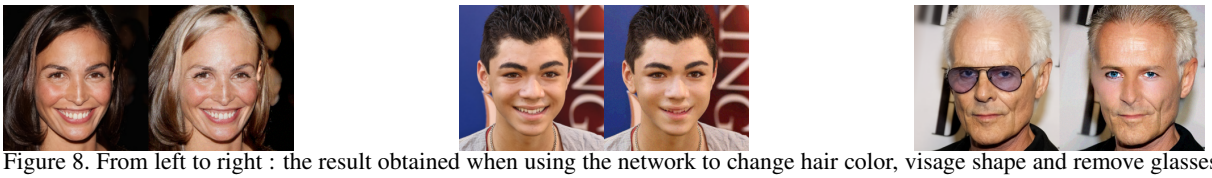

Figure 8. From left to right : the result obtained when using the network to change hair color, visage shape and remove glasses. For every example the original image is shown on the left.

	Emerging	3×3 convolution
Affine coupling	3.3851	3.4209
Quad-coupling	3.3612	3.3879

Table 3. Comparison of the performance (in bits per dimension) achieved on the Cifar10 dataset with different architectures.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Figure 9. Here we can see the result of gradually modifying the age parameter. The original image is the fourth from the left.