future internet

MDPI

Article

Fast Flow Reconstruction via Robust Invertible $n \times n$ Convolution

Thanh-Dat Truong [1,*], Chi Nhan Duong [2], Minh-Triet Tran [3], Ngan Le [1] and Khoa Luu [1]

[1] Computer Vision and Image Understanding Lab, Department of Computer Science and Computer Engineering, University of Arkansas, Fayetteville, AR 72501, USA; thile@uark.edu (N.L.); khoaluu@uark.edu (K.L.)
[2] Department of Computer Science and Software Engineering, Concordia University, Montreal, QC H3G 2V4, Canada; dcnhan@ieee.org
[3] Faculty of Information Technology, University of Science, VNU-HCM, Ho Chi Minh 721337, Vietnam; tmtriet@hcmus.edu.vn
* Correspondence: tt032@uark.edu; Tel.: +1-479-404-0772

Abstract: Flow-based generative models have recently become one of the most efficient approaches to model data generation. Indeed, they are constructed with a sequence of invertible and tractable transformations. Glow first introduced a simple type of generative flow using an invertible 1×1 convolution. However, the 1×1 convolution suffers from limited flexibility compared to the standard convolutions. In this paper, we propose a novel invertible $n \times n$ convolution approach that overcomes the limitations of the invertible 1×1 convolution. In addition, our proposed network is not only tractable and invertible but also uses fewer parameters than standard convolutions. The experiments on CIFAR-10, ImageNet and Celeb-HQ datasets, have shown that our invertible $n \times n$ convolution helps to improve the performance of generative models significantly.

Keywords: flow-based generative model; invertible $n \times n$ convolution; invertible and tractable transformations

Citation: Truong, T.-D.; Duong, C.N.; Tran, M.-T.; Le, N.; Luu, K. Fast Flow Reconstruction via Robust Invertible $n \times n$ Convolution. *Future Internet* **2021**, *13*, 179. https://doi.org/10.3390/fi13070179

Academic Editor: Massimo Cafaro

Received: 31 May 2021
Accepted: 6 July 2021
Published: 8 July 2021

1. Introduction

Supervised deep learning models have recently achieved numerous breakthrough results in various applications, for example, Image Classification [1–3], Object Detection [4–6], Face Recognition [7–14], Image Segmentation [15,16] and Generative Model [17–22]. However, these methods usually require a huge number of annotated data, which is highly expensive. In order to tackle the requirement of large annotations, generative models have become a feasible solution. The main objective of generative models is to learn the hidden dependencies that exist in the realistic data so that they can extract meaningful features and variable interactions to synthesize new realistic samples without human supervision or labeling. Generative models can be used in numerous applications such as anomaly detection [23], image inpainting [24], data generation [20,25], super-resolution [26], face synthesis [22,27,28], and so forth. However, learning generative models is an extremely challenging process due to high-dimensional data.

There are two types of generative models extensively deployed in recent years, including likelihood-based methods [29–32] and Generative Adversarial Networks (GANs) [33]. Likelihood-based methods have three main categories: Autoregressive models [30], variational autoencoders (VAEs) [34], and flow-based models [29,31,32]. The flow-based generative model is constructed using a sequence of invertible and tractable transformations, the model explicitly learns the data distribution and therefore the loss function is simply a negative log-likelihood.

The flow-based model was first introduced in [31] and later extended in RealNVP [32]. These methods introduced an affine coupling layer that is invertible and tractable based on Jacobian determinant. As the design of the coupling layers, at each stage, only a subset of data is transformed while the rest is required to be fixed. Therefore, they may be

limited in flexibility. To overcome this limitation, coupling layers are alternated with less complex transformations that manipulate on all dimensions of the data. In RealNVP[32], the authors use a fixed channel permutation using fixed checkerboard and channel-wise masks. Kingma et al. [29] simplifies the architecture by replacing the reverse permutation operation on the channel ordering with invertible 1×1 convolutions.

However, the 1×1 convolutions are not flexible enough in these scenarios. It is extremely hard to compute the inverse form of the standard $n \times n$ convolutions, and this step usually produces high computational costs. There are prior approaches that design the invertible $n \times n$ convolutions by using emerging convolution [35], periodic convolutions [35], autoregressive flow [36] or stochastic approximation [37–39]. In this paper, we propose an approach to generalize an invertible 1×1 convolution to a more general form of $n \times n$ convolution. Firstly, we reformulate the standard convolution layer by shifting the inputs instead of the kernels. Then, we propose an invertible shift function that is a tractable form of Jacobian determinant. Through the experiments on CIFAR-10 [40], ImageNet [41] and Celeb-HQ [42] datasets, we prove that our proposals are significant and efficient for high-dimensional data. Figure 1 illustrates the advantages of our approach with high-resolution synthesized images.

Figure 1. Reconstruction results using our proposed approach.

Contributions: This work generalizes the invertible 1×1 convolution to an invertible $n \times n$ convolution by reformulating the convolution layer using our proposed invertible shift function. Our contributions can be summarized as follows:

- Firstly, by analyzing the standard convolution layer, we reformulate its equation into a form such that, rather than shifting the kernels during the convolution process, shifting the input provides equivalent results.
- Secondly, we propose a novel invertible shift function that mathematically helps to reduce the computational cost of the standard convolution while keeping the range of the receptive fields. The determinant of the Jacobian matrix produced by this shift function can be computed efficiently.
- Thirdly, evaluations of several datasets on both objects and faces have shown the generalization of the proposed $n \times n$ convolution using our proposed novel invertible shift function.

2. Related Work

The generative models can be divided into two groups, that is, *Generative Adversarial Networks* and *Flow-based Generative Models*. In the first group, *Generative Adversarial Networks* [33] provide an appropriate solution to model the data generation. The discriminative model learns to distinguish the real data from the fake samples produced using a generative model. Two models are trained as they are playing a mini-max game. Meanwhile, in the second group, the *Flow-based Generative Models* [29,31,32] are constructed using a

sequence of ***invertible*** and ***tractable*** transformations. Unlike GAN, the model explicitly learns the data distribution $p(\mathbf{x})$ and therefore the loss function is efficiently employed with the log-likelihood.

In this section, we discuss several types of flow-based layers that are commonly used in flow-based generative models. An overview of several invertible functions is provided in the Table 1. In particular, all functions easily obtain the reverse function and tractability of a Jacobian determinant. The symbols $\odot$, / denote element-wise multiplication and division. h, w denotes the height and width of the input/output. c, i, j are the depth channel index and spatial indices, respectively.

Table 1. Comparative invertible functions in several generative normalizing flows.

Description	Function	Reverse Function	Log-Determinant
ActNorm [29]	$\mathbf{y} = \mathbf{x} \odot \gamma + \beta$	$\mathbf{x} = (\mathbf{y} - \beta)/\gamma$	$\sum \log \lvert\gamma\rvert$
Affine Coupling [32]	$\mathbf{x} = [\mathbf{x}_a, \mathbf{x}_b]$ $\mathbf{y}_a = \mathbf{x}_a \odot s(\mathbf{x}_b) + t(\mathbf{x}_b)$ $\mathbf{y} = [\mathbf{y}_a \mathbf{x}_b]$	$\mathbf{y} = [\mathbf{y}_a, \mathbf{y}_b]$ $\mathbf{x}_a = [\mathbf{y}_a - t(\mathbf{y}_b)]/s(\mathbf{y}_b)$ $\mathbf{x} = [\mathbf{x}_a \mathbf{y}_b]$	$\sum \log \lvert s(\mathbf{x}_b)\rvert$
1×1 conv [29]	$\mathbf{y}_{:,i,j} = \mathbf{W}\mathbf{x}_{:,i,j}$	$\mathbf{x}_{:,i,j} = \mathbf{W}^{-1}\mathbf{y}_{:,i,j}$	$h.w. \log \lvert \det \mathbf{W}\rvert$
Our Shift Function	$\mathbf{y}_{c,i,j} = \alpha_c \mathbf{x}_{c,i,j} + \beta_c$	$\mathbf{x}_{c,i,j} = [\mathbf{y}_{c,i,j} - \beta_c]/\alpha_c$	$h.w. \sum_c \log \lvert\alpha_c\rvert$

Coupling Layers: NICE [31] and RealNVP [32] presented coupling layers with a normalizing flow by stacking a sequence of invertible bijective transformation functions. The bijective function is designed as an affine coupling layer, which is a tractable form of Jacobian determinant. RealNVP can work in a multi-scale architecture to build a more efficient model for large inputs. To further improve the propagation step, the authors applied batch normalization and weight normalization during training. Later, Ho et. al. [43] presented a continuous mixture cumulative distribution function to improve the density modeling of coupling layers. In addition to improving the expressiveness of transformations of coupling layers [43], utilized multi-head self-attention layers [44] in the transformations.

Inverse Autoregressive Convolution: Germain et al. [45] introduced autoregressive autoencoders by constructing an extension of a non-variational autoencoder that can estimate distributions and is straightforward in computing their Jacobian determinant. Masked autoregressive flow [36] is a type of normalizing flow, where the transformation layer is built as an autoregressive neural network. Inverse autoregressive flow [30] formulates the conditional probability of the target variable as an autoregressive model.

Invertible 1×1 Convolution: Kingma et al. [29] proposed simplifying the architecture via invertible 1×1 convolutions. Learning a permutation matrix is a discrete optimization that is not amenable to gradient ascent. However, the permutation operation is simply a special case of a linear transformation with a square matrix. We can pursue this work with convolutional neural networks, as permuting the channels is equivalent to a 1×1 convolution operation with an equal number of input and output channels. Therefore, the authors replace the fixed permutation with learned 1×1 convolution operations.

Activation Normalization: [29] performs an affine transformation using scale and bias parameters per channel. This layer simply shifts and scales the activations with data-dependent initialization that normalizes the activations given an initial minibatch of data. This allows the scaling down of the minibatch size to 1 (for large images) and the scaling up of the size of the model.

Invertible $n \times n$ Convolution: Since the invertible 1×1 convolution is not flexible, Hoogeboom et al. [35] proposed an invertible $n \times n$ convolution generalized from the 1×1 convolutions. The authors presented two methods to produce the invertible convolutions: (1) *Emerging Convolution* and (2) *Invertible Periodic Convolutions*. Emerging Convolution is obtained by chaining specific invertible autoregressive convolutions [30] and speeding up this layer through the use of an accelerated parallel inversion module implemented

in Cython. Invertible Periodic Convolutions transform data to the frequency domain via Fourier transform; this alternative convolution has a tractable Jacobian determinant and inverse. However, these invertible $n \times n$ convolutions require more parameters; therefore, these have an additional computational cost compared to our proposed method.

Lipschitz Constant: Behrmann et al. [37] developed a theory that any residual blocks satisfying the Lipschitz Constant can be invertible. Hence, Behrmann et al. proposed an invertible residual network (i-ResNet) as a normalizing flow-based model. Similar to [29,31,32,35], i-ResNet is learned by optimizing the negative log-likelihood in which the inverse flow and Jacobian determinant of the residual block can be efficiently approximated by the stochastic methods. Inheriting the success of Lipschitz theory, Kim et al. [38] proposed an L_2 self-attention that allows the self-attention of the Transformer networks [44] to be invertible.

3. Background

3.1. Flow-Based Generative Model

Let $\mathbf{x}$ be a high-dimensional vector with unknown true distribution $\mathbf{x} \sim p_{\mathcal{X}}(\mathbf{x})$, $x \in \mathcal{X}$, a simple prior probability distribution $p_{\mathcal{Z}}$ on a latent variable $z \in \mathcal{Z}$, a bijection $f : \mathcal{X} \to \mathcal{Z}$, the change of variable formula defines a model distribution on $\mathcal{X}$ as shown in Equation (1).

$$p_{\mathcal{X}}(\mathbf{x}) = p_{\mathcal{Z}}(\mathbf{z}) \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|, \tag{1}$$

where $\frac{\partial f(x)}{\partial x}$ is the Jacobian of f at $\mathbf{x}$. The log-likelihood objective is then equivalent to minimizing:

$$\begin{aligned} \mathcal{L}(\mathcal{X}) &= -_{\mathbf{x} \in \mathcal{X}} \log p_{\mathcal{X}}(\mathbf{x}) \\ &= -_{\mathbf{x} \in \mathcal{X}} \left[\log p_{\mathcal{Z}}(\mathbf{z}) + \log \left| \det \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| \right]. \end{aligned} \tag{2}$$

Since the data $\mathbf{x}$ are discrete data, we add a random uniform noise $u \in \mathcal{U}(0, a)$, where a is determined by the discretization level of the data, to make $\mathbf{x}$ be continuous data. The generative process can be defined as Equation (3).

$$\begin{aligned} \mathbf{z} &\sim p_{\mathcal{Z}}(\mathbf{z}) \\ \mathbf{x} &= f^{-1}(\mathbf{z}). \end{aligned} \tag{3}$$

The bijection function f is constructed from a sequence of invertible and tractable Jacobian determinant transformations: $f = f_1 \circ f_2 \circ ... \circ f_K$ (K is the number of transformations). Such a sequence of invertible transformations is also called a normalizing flow. Here, Equation (2) can be written as in Equation (4).

$$\begin{aligned} \mathcal{L}(\mathcal{X}) &= -_{\mathbf{x} \in \mathcal{X}} \log p_{\mathcal{X}}(\mathbf{x}) \\ &= -_{\mathbf{x} \in \mathcal{X}} \left[\log p_{\mathcal{Z}}(\mathbf{z}) + \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}} \right) \right| \right] \end{aligned} \tag{4}$$

where $\mathbf{h}_k = f_1 \circ f_2 \circ .. \circ f_k(\mathbf{h}_0)$ with $\mathbf{h}_0 = \mathbf{x}$.

3.2. Standard $n \times n$ Convolution

In this section, we revisit the standard $n \times n$ convolution. Let $\mathbf{X}$ be an $C \times H \times W$ input; $\mathbf{W}$ is a $D \times C \times K$ kernel, and the convolution can be expressed as follows:

$$\begin{aligned} \mathbf{Y} = \mathbf{W} \star \mathbf{X} &= \begin{bmatrix} \mathbf{W}_{:,:,1} & \mathbf{W}_{:,:,2} & \cdots & \mathbf{W}_{:,:,k} \end{bmatrix} \times \begin{bmatrix} \mathbf{X}^1_{:,:,:} \\ \mathbf{X}^2_{:,:,:} \\ \vdots \\ \mathbf{X}^K_{:,:,:} \end{bmatrix} \\ &= \sum_{k=1}^{K} \mathbf{W}_{:,:,k} \times \mathbf{X}^k_{:,:,:} = \sum_{k=1}^{K} \mathbf{W}_{:,:,k} \times \mathcal{S}_k(\mathbf{X}), \end{aligned} \tag{5}$$

where $\mathbf{X}^k_{:,:,:}$ is a $C \times H \times W$ matrix that represents a spatially shifted version of input matrix $\mathbf{X}$ with shift amount (i_k, j_k), . $\mathbf{W}_{:,:,k}$ represents the $D \times C$ matrix corresponding to the kernel index k, the symbol $\star$ denotes a convolution operator.

In Equation (5), the standard convolution is simply a sum of 1×1 convolutions on shifted inputs. The function $\mathcal{S}_k$ maps the input $\mathbf{X}$ to the corresponding shifted input $\mathbf{X}^k_{:,:,:}$. The standard convolution uses the common shifted input with integer-valued shift amounts for index k. Figure 2 illustrates our reformulated $n \times n$ convolution, if we can share the shifted inputs regardless of the kernel index, especially $\mathcal{S}_k(\mathbf{X}) = \mathcal{S}(\mathbf{X})$, the standard convolution will be simplified as the 1×1 convolution as shown in Equation 6. In this paper, we propose a shift function $\mathcal{S}$, which is an invertible and tractable form of the Jacobian determinant.

$$\begin{aligned} \sum_{k=1}^{K} \mathbf{W}_{:,:,k} \times \mathcal{S}_k(\mathbf{X}) &= \sum_{k=1}^{K} \mathbf{W}_{:,:,k} \times \mathcal{S}(\mathbf{X}) \\ &= \left(\sum_{k=1}^{K} \mathbf{W}_{:,:,k} \right) \times \mathcal{S}(\mathbf{X}). \end{aligned} \tag{6}$$

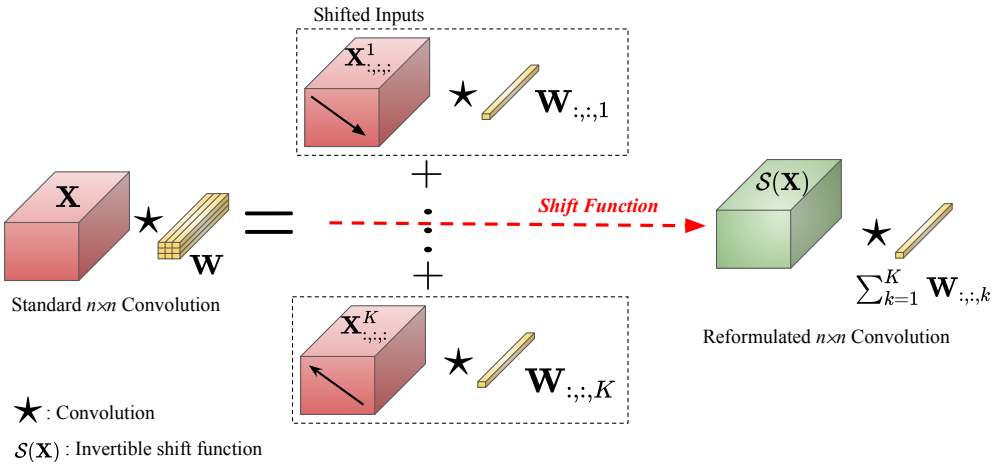


Figure 2. Reformulating $n \times n$ convolution. We propose to shift inputs instead of kernels. The proposed invertible $n \times n$ convolution will be simplified as a combination of the invertible shift function $\mathcal{S}$ and the invertible 1×1 convolution.

4. Invertible $n \times n$ Convolution

In this section, we first introduce our proposed Invertible Shift Function and then present invertible $n \times n$ convolution in details.

4.1. Invertible Shift Function

The shift function $\mathcal{S}$ will approximate all shifted input $\mathbf{X}^{k}_{:,:,:}$ $(1 \leq k \leq K)$. Here, we propose to design $\mathcal{S}$ as a linear transformation per channel; specifically, we have learnable variables α_c, β_c; $1 \leq c \leq C$ are scale and translation parameters for each channel, respectively. The shift function $\mathcal{S}$ can be formulated as follows:

$$\mathcal{S}(\mathbf{X}_{c,i,j}) = \alpha_c \mathbf{X}_{c,i,j} + \beta_c, \tag{7}$$

where c, i, j are the depth channel index and spatial indices, respectively. The reverse function of $\mathcal{S}$ can be easy to obtain:

$$\mathbf{X}_{c,i,j} = \frac{\mathcal{S}(\mathbf{X}_{c,i,j}) - \beta_c}{\alpha_c}. \tag{8}$$

Thanks to Equation (7), the value of $\mathcal{S}(\mathbf{X}_{c,i,j})$ only depends on $\mathbf{X}_{c,i,j}$ and the Jacobian matrix will be in the form of the diagonal matrix as follows:

$$\begin{aligned} \mathbf{J} = \frac{\partial \mathcal{S}(\mathbf{X})}{\partial \mathbf{X}} &= \begin{bmatrix} \frac{\partial \mathcal{S}(\mathbf{X}_{1,1,1})}{\partial \mathbf{X}_{1,1,1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial \mathcal{S}(\mathbf{X}_{1,1,2})}{\partial \mathbf{X}_{1,1,2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \mathcal{S}(\mathbf{X}_{C,H,W})}{\partial \mathbf{X}_{C,H,W}} \end{bmatrix} \\ &= \begin{bmatrix} \alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_c. \end{bmatrix} \end{aligned} \tag{9}$$

Therefore, the determinant of Equation (9) is the product of all elements in the diagonal of the matrix $\mathbf{J}$ as in Equation (10).

$$\begin{aligned} \det\left(\frac{\partial \mathcal{S}(\mathbf{X})}{\partial \mathbf{X}}\right) &= \prod_{c=1}^{C} \alpha_c^{H \times W} \\ \log\left|\det\left(\frac{\partial \mathcal{S}(\mathbf{X})}{\partial \mathbf{X}}\right)\right| &= H \times W \times \sum_{c=1}^{C} \log|\alpha_c|. \end{aligned} \tag{10}$$

4.2. Invertible $n \times n$ Convolution

Kingma [29] proposed invertible 1×1 convolution as the smart way to learn the permutation matrix instead of the fixed permutation [31,32]. However, the 1×1 suffers from limited flexibility compared to the standard convolution. In particular, the receptive fields of 1×1 convolution is limited. When the network goes deeper, the receptive fields of 1×1 convolutions are still small areas; these, therefore, cannot generalize or model large objects of high-dimensional data. However, the 1×1 convolution has its own advantages compared to the standard convolution. First, the 1×1 convolution allows the network to compress the data of the input volume to be smaller. Second, 1×1 suffers less over-fitting due to small kernel sizes. Therefore, in our proposal, we still take advantages of the 1×1 convolution. Specifically, we adopt the successfully invertible 1×1 convolution of Glow [29] in our design.

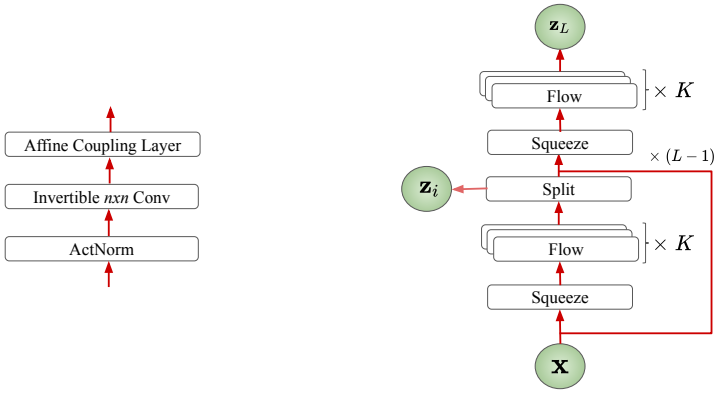
In the previous subsection, we proved that the shift function $\mathcal{S}$ is invertible and proved the tractability of the Jacobian determinant. In Section 3.2, we indicated that if we can share shifted inputs regardless of the kernel index via the shift function $\mathcal{S}$, we can simplify the standard $n \times n$ convolution to the composition of the $\mathcal{S}$ and 1×1 convolution. Therefore, the invertible $n \times n$ convolution will be equivalent to the combination of the invertible shift function $\mathcal{S}$ and the invertible 1×1 convolution. Specifically, the input will first be

forwarded to the shift function $\mathcal{S}$ and then convoluted with the 1×1 filter. Algorithm 1 illustrates the pseudo code of the invertible $n \times n$ convolution.

Algorithm 1: Invertible $n \times n$ Convolution

Input: An input $\mathbf{X} \in \mathbb{R}^{N \times H \times W \times C}$
Result: An output of invertible $n \times n$ convolution and the log Jacobian determinant
Initialize $\alpha, \beta \in \mathbb{R}^{C}$ for the invertible shift function;
Initialize $\mathbf{W} \in \mathbb{R}^{C \times C}$ as a rotation matrix for the invertible 1×1 convolution function;
logdet = 0.0;
The invertible shift function;
$\mathbf{Y} = \mathbf{X} \times \alpha + \beta$ (Channel-wise operations);
The inverse will be $\mathbf{X} = \frac{\mathbf{Y} - \beta}{\alpha}$;
logdet = logdet + $\sum_{i=1}^{C} \log(\alpha_i)$;
The invertible 1×1 ***convolution***;
$\mathbf{Z} = Conv(\mathbf{Y}, \mathbf{W})$;
The inverse will be $\mathbf{Y} = Conv(\mathbf{Z}, \mathbf{W}^{-1})$;
logdet = logdet + $\log(\det(\mathbf{W}))$ * H * W;
Return **Z** and logdet;

Figure 3a illustrates our one step of flow. We adopt the common design of a flow step [29,35,46] in our design. Our proposal can be easily integrated to the multi-scale architecture designed by Dinh et al. [32] (Figure 3b). From our proposal, we can generalize the invertible 1×1 convolution to the invertible $n \times n$ convolution through the shift function $\mathcal{S}$. It can help to encourage the filters to learn a more efficient data representation and embed more useful latent features than the invertible 1×1 convolution used in Glow [29]. Besides, we use fewer parameters and have less inference time compared to the standard $n \times n$ convolutions.



(a) One step of flow (b) Multi-scale Architecture

Figure 3. (**a**) is our one step of flow using an invertible $n \times n$ convolution. Our proposal flow step is able to combine with the multi-scale architecture designed in RealNVP (**b**). K and L are the depth of flow and the number of levels, respectively.

5. Experiments

In this section, we present our experimental results on CIFAR-10, ImageNet and Celeb-HQ datasets. Firstly, in Section 5.1, we compare log-likelihood against the previous flow-based models, that is, RealNVP [32], Glow [29] and Emerging Convolution [35]. Finally, in Section 5.2, we show our qualitative results trained on the Celeb-HQ dataset.

5.1. Quantitative Experiments

Datasets and Metric: We evaluate our invertible $n \times n$ convolution on CIFAR-10 (Figure 4a) and ImageNet (Figure 4b) with 32×32 and 64×64 image sizes. We use bits per dimension as the criteria with which to evaluate models. We compare our method against RealNVP [32], Glow [29] and Emerging Convolution [35]. We adopt the network structures of Glow and replace all invertible 1×1 convolutions of Glow with our invertible $n \times n$ convolutions. For the data preprocessing, we follow the same process as in RealNVP [32].

(a) CIFAR-10 Dataset (b) ImageNet Dataset (c) Celeb-HQ Dataset

Figure 4. The examples from the CIFAR dataset (**a**), ImageNet dataset (**b**) and Celeb-HQ dataset (**c**).

Network Configurations: In the CIFAR experiment, the depth of flow K and the number of levels L are set to 32 and 3, respectively. Meanwhile, the depth of flow in ImageNet experiments is set to 48, the numbers of levels of ImageNet 32×32 and ImageNet 64×64 experiments are set to 3 and 4, respectively. We use the Adam optimizer [47] to optimize the networks in which batch size and learning rate are set to 64 (per GPU) and 0.001, respectively. We choose Normal Distribution as the prior distribution $p_{\mathcal{Z}}(z) \sim \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$ in all experiments.

The shift function $\mathcal{S}$ will be not inverse if the $\alpha_c = 0$ ($\exists\ c \in [1...C]$). Hence, in the training process, we will first initialize $\alpha_c = 1$ and $\beta_c = 0$ ($1 \leq c \leq C$). During the learning processing, we keep α_c ($1 \leq c \leq C$) as a different 0 to guarantee that the shift function $\mathcal{S}$ is inverse and to guarantee the tractability of the Jacobian determinant. Training models on high-dimensional data requires large memory. To be able to train with a large batch size, we simultaneously and distributively trained the models on four GPUs via Horovod (https://github.com/horovod/horovod, accessed on 8 July 2021) and TensorFlow (https://tensorflow.org, accessed on 8 July 2021) frameworks.

Results: Table 2 shows our experimental results. In particular, our proposal helps to improve the generative models on ImageNet 32×32 and ImageNet 64×64 datasets, which are more challenging than CIFAR-10. In particular, our proposed method achieves a state-of-the-art performance in which the bit per dimension results in ImageNet 32×32, and ImageNet 64×64 is **3.96** and **3.74**, respectively. In comparison, the Emerging Convolution [35] and Glow achieve similar results in both ImageNet 32×32 and ImageNet 64×64 benchmarks, which are 4.09 and 3.81, respectively. Meanwhile, the corresponding results of RealNVP on these benchmarks are 4.28 and 3.98, respectively. As shown by the results, our proposed invertible $n \times n$ convolution provides a better generative capability than the stand-alone invertible 1×1 convolution. Since Emerging Convolution uses invertible auto-regressive convolution, our proposal is, therefore, less complicated and has faster inference than Emerging Convolution. In the CIFAR-10 benchmark, although our model does not perform as well as Glow [29] and Emerging Convolution [35], we find it interesting that our method gains competitive results with a small number of modifications. The gap in performance is partially caused by the small amount of CIFAR-10 data that is inefficient for training the well-generalized convolution.

Table 2. Comparative results (bits per dimension) of proposed invertible $n \times n$ convolution compared to RealNVP, Glow and Emerging Convolution.

Models	CIFAR-10	ImageNet 32	ImageNet 64
RealNVP	3.49	4.28	3.98
Glow	3.35	4.09	3.81
Emerging Conv	**3.34**	4.09	3.81
Ours	3.50	**3.96**	**3.74**

5.2. Qualitative Experiments

The CelebA-HQ dataset [42] was selected to train the model using the architectures defined in the previous section with a higher resolution (256×256 image sizes). The depth of flow K and the number of levels L were set to 32 and 6, respectively. Since high-dimensional data requires large memory, we reduced the batch size to 1 (per GPU) and trained on eight GPUs. The qualitative experiment aims to study the efficiency of the model when it scales up to the high-resolution images, synthesizes realistic images, and provides the meaningful latent space. Figure 4c shows the examples of Celeb-HQ datasets. We trained our model on 5-bit images in order to improve visual quality with a slight trade-off of color fidelity. As shown by the synthetic images in Figure 5, our model can generalize realistic images in high dimensional data.

Figure 5. Synthetic celebrity faces sampled from our model trained on the CelabA-HQ dataset.

6. Conclusions and Future Work

This paper has presented a novel invertible $n \times n$ convolution approach. By reformulating the convolution layer, we propose to use the shift function to shift inputs instead of kernels. We prove that our shift function is invertible and tractable in terms of calculating the Jacobian determinant. The method leverages the shift function and the invertible 1×1 convolution to generalize to the invertible $n \times n$ convolution. Through experiments, our proposal has achieved state-of-the-art results in quantitative measurement and is able to generate realistic images with high-resolution.

There are several challenges that remain to be addressed in future work. In particular, when the model scales up to the high-resolution images, it requires a large amount of GPU memory during the training process, that is, the back-propagation process. Maintaining the rotation matrix property for the invertible 1×1 convolution when training the model on a large dataset is also a challenging task, since the model easily falls into the non-inverse matrix due to the stochastic gradient update of the back-propagation algorithm. That issue is interesting work and should be improved in the future.

Author Contributions: Conceptualization: T.-D.T. and C.N.D. Methodology: T.-D.T., C.N.D. and K.L. Review and Editing: K.L., M.-T.T., and N.L. Supervision: K.L. and M.-T.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Science Foundation (NSF).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: CIFAR Dataset https://www.cs.toronto.edu/~kriz/cifar.html, accessed on 8 July 2021, ImageNet dataset https://image-net.org/, accessed on 8 July 2021, and CelebA-HQ Dataset https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html, accessed on 8 July 2021.

Acknowledgments: This work is partially supported by NSF EPSCoR Track-1 Data Science, Data Analytics that are Robust and Trusted (DART), NSF Track-2 CRESH, and NSF 19-554 Small Business Innovation Research Program. The authors would like to thank the reviewers for their valuable comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
2. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
3. Sun, S.; Pang, J.; Shi, J.; Yi, S.; Ouyang, W. FishNet: A Versatile Backbone for Image, Region, and Pixel Level Prediction. Available online: https://arxiv.org/abs/1901.03495 (accessed on 8 July 2021).
4. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *Lecture Notes in Computer Science Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
5. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
6. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
7. Luu, K.; Seshadri, K.; Savvides, M.; Bui, T.; Suen, C. Contourlet Appearance Model for Facial Age Estimation. In Proceedings of the 2011 International Joint Conference on Biometrics (IJCB), Washington, DC, USA, 11–13 October 2011.
8. Le, H.; Seshadri, K.; Luu, K.; Savvides, M. Facial Aging and Asymmetry Decomposition Based Approaches to Identification of Twins. *Pattern Recognit.* **2015**, *48*, 3843–3856.
9. Xu, F.; Luu, K.; Savvides, M. Spartans: Single-sample Periocular-based Alignment-robust Recognition Technique Applied to Non-frontal Scenarios. *IEEE Trans. Image Process.***2015**, *12*, 4780–4795, doi: 10.1109/TIP.2015.2468173.
10. Xu, J.; Luu, K.; Savvides, M.; Bui, T.; Suen, C. Investigating Age Invariant Face Recognition Based on Periocular Biometrics. In Proceedings of the 2011 International Joint Conference on Biometrics (IJCB), Washington, DC, USA, 11–13 October 2011.
11. Duong, C.; Quach, K.; Luu, K.; Le, H.K., Jr. Fine Tuning Age Estimation with Global and Local Facial Features. In Proceedings of the 36th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Prague, Czech Republic, 22–27 May 2011.
12. Luu, K.; Bui, T.K., Jr.; Suen, C. Age Estimation using Active Appearance Models and Support Vector Machine Regression. In Proceedings of the 2009 IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems, Washington, DC, USA, 28–30 September 2009.
13. Luu, K.; Bui, T.; Suen, C. Kernel Spectral Regression of Perceived Age from Hybrid Facial Features. In Proceedings of the 2011 IEEE International Conference on Automatic Face and Gesture Recognition (FG), Santa Barbara, CA, USA, 21–25 March 2011.
14. Chen, C.; Yang, W.; Wang, Y.; Ricanek, K.; Luu, K. Facial Feature Fusion and Model Selection for Age Estimation. In Proceedings of the 2011 IEEE International Conference on Automatic Face and Gesture Recognition (FG), Santa Barbara, CA, USA, 21–25 March 2011.

15. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 834–848.
16. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
17. Luu, K.K., Jr.; Bui, T.; Suen, C. The Familial Face Database: A Longitudinal Study of Family-based Growth and Development on Face Recognition. In Proceedings of the Robust Biometrics: Understanding Science and Technology, Marriott Waikiki, HI, USA, 2–5 November 2008.
18. Luu, K. Computer Approaches for Face Aging Problems. In Proceedings of the 23th Canadian Conference On Artificial Intelligence (CAI), Ottawa, ON, Canada, 31 May–2 June 2010
19. Mirza, M.; Osindero, S. Conditional generative adversarial nets. *arXiv* **2014**, arXiv:1411.1784
20. Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv* **2017**, arXiv:1710.10196
21. Duong, C.; Luu, K.; Quach, K.; Bui, T. Longitudinal Face Modeling via Temporal Deep Restricted Boltzmann Machines. In Proceedings of the 2016 IEEE Conference On Computer Vision And Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
22. Duong, C.; Quach, K.; Luu, K.; Le T.; Savvides, M. Temporal Non-volume Preserving Approach to Facial Age-Progression and Age-Invariant Face Recognition. In Proceedings of the 2017 IEEE International Conference On Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
23. Mattia, F.D.; Galeone, P.; Simoni, M.D.; Ghelfi, E. A Survey on GANs for Anomaly Detection. *arXiv* **2019**, arXiv:1906.11632.
24. Yu, J.; Lin, Z.; Yang, J.; Shen, X.; Lu, X.; Huang, T.S. Generative image inpainting with contextual attention. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5505–5514.
25. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
26. Ledig, C.; Theis, L.; Huszár, F.; Caballero, J.; Cunningham, A.; Acosta, A.; Aitken, A.; Tejani, A.; Totz, J.; Wang, Z.; et al. Photo-realistic single image super-resolution using a generative adversarial network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4681–4690.
27. Duong, C.; Luu, K.; Quach, K.; Nguyen, N.; Patterson, E.; Bui, T.; Le N. Automatic Face Aging in Videos via Deep Reinforcement Learning. In Proceedings of the 2019 IEEE/CVF Conference On Computer Vision And Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
28. Duong, C.; Luu, K.; Quach, K.; Bui, T. Deep Appearance Models: A Deep Boltzmann Machine Approach for Face Modeling. *Int. J. Comput. Vis.***2019**, *127*, 437–455. doi: 10.1007/s11263-018-1113-3.
29. Kingma, D.P.; Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 10215–10224.
30. Kingma, D.P.; Salimans, T.; Jozefowicz, R.; Chen, X.; Sutskever, I.; Welling, M. Improved Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems 29*; Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2016; pp. 4743–4751.
31. Dinh, L.; Krueger, D.; Bengio, Y. NICE: Non-linear Independent Components Estimation. *arXiv* 2015, arXiv:1410.8516.
32. Dinh, L.; Sohl-Dickstein, J.; Bengio, S. Density estimation using Real NVP. In Proceedings of the 3rd International Conference on Learning Representations, ICLR, Toulon, France, 24–26 April 2017.
33. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2014; pp. 2672–2680.v
34. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. In Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, 14–16 April 2014.
35. Hoogeboom, E.; van den Berg, R.; Welling, M. Emerging Convolutions for Generative Normalizing Flows. *arxiv* **2019**, arxiv: 1901.11137.
36. Papamakarios, G.; Murray, I.; Pavlakou, T. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems 30*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds., Curran Associates, Inc.: Red Hook, NY, USA, 2017; pp. 2335–2344.
37. Behrmann, J.; Grathwohl, W.; Chen, R.T.Q.; Duvenaud, D.; Jacobsen, J.H. Invertible Residual Networks. In Proceedings of the 36th International Conference on Machine Learning, Beach, CA, USA, 10–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; PMLR: Long Beach, CA, USA, 2019; Volume 97; pp. 573–582.
38. Kim, H.; Papamakarios, G.; Mnih, A. The Lipschitz Constant of Self-Attention. *arXiv* **2021**, arXiv:2006.04710
39. Chen, R.T.; Behrmann, J.; Duvenaud, D.; Jacobsen, J.H. Residual flows for invertible generative modeling. *arXiv* **2019**, arXiv:1906.02735.
40. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. 2009. Available online: http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 8 July 2021).
41. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009.

42. Liu, Z.; Luo, P.; Wang, X.; Tang, X. Deep Learning Face Attributes in the Wild. In Proceedings of International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015.
43. Ho, J.; Chen, X.; Srinivas, A.; Duan, Y.; Abbeel, P. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. *arXiv* **2019**, arXiv:1902.00275
44. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.U.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
45. Germain, M.; Gregor, K.; Murray, I.; Larochelle, H. MADE: Masked Autoencoder for Distribution Estimation. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Bach, F.; Blei, D., Eds.; PMLR: Lille, France, 2015; Volume 37, pp. 881–889.
46. Truong, D.; Nhan, Duong, C.; Luu, K.; Tran, M.; Le N. Domain Generalization via Universal Non-volume Preserving Approach. In Proceedings of the 2020 17th Conference On Computer And Robot Vision (CRV), Ottawa, ON, Canada, 13–15 May 2020.
47. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.