Supplementary Material for iVPF: Numerical Invertible Volume Preserving Flow for Efficient Lossless Compression

Shifeng Zhang Chen Zhang Ning Kang Zhenguo Li
Huawei Noah's Ark Lab
{zhangshifeng4, chenzhang10, kang.ning2, li.zhenguo}@huawei.com

A. Detailed Implementation of iVPF

A.1. Normalising Flow and Volume Preserving Flows

Unless specified, in this subsection, we discuss the continuous flow, which is generally used for training. Note that iVPF is constructed based on the trained model with minor modifications.

A.1.1 Normalising Flows

As discussed in Sec. 3.1, the general normalising flow establishes a continuous bijection $f : \mathcal{X} \to \mathcal{Z}$ between input data $\mathcal{X}$ and latent space $\mathcal{Z}$. The flow model is constructed with composition of flow layers such that $f = f_L \circ ... \circ f_2 \circ f_1$. Let $\mathbf{y}_l = f_l(\mathbf{y}_{l-1}), \mathbf{y}_0 = \mathbf{x}$ and $\mathbf{z} = \mathbf{y}_L$. The prior distribution $p_Z(\mathbf{z})$ can be stipulated from any parametric distribution families, e.g., Gaussians, logistic distribution, etc. Then the model distribution $p_X(\mathbf{x})$ of input data $\mathbf{x}$ can be expressed with the latent variable $\mathbf{z}$ such that

$$\log p_X(\mathbf{x}) = \log p_Z(\mathbf{z}) + \sum_{i=1}^{L} \log |\frac{\partial \mathbf{y}_l}{\partial \mathbf{y}_{l-1}}|, \tag{1}$$

where $|\frac{\partial \mathbf{y}_l}{\partial \mathbf{y}_{l-1}}|$ is the absolute value of the determinant of the Jacobian matrix.

The inverse flow is $f^{-1} = f_1^{-1} \circ ... \circ f_L^{-1}$. Given latents $\mathbf{y}_L = \mathbf{z}$, the original data $\mathbf{x}$ can be recovered with $\mathbf{y}_{l-1} = f_l^{-1}(\mathbf{y}_l)$ and $\mathbf{x} = \mathbf{y}_0$.

A.1.2 Constructing Volume Preserving Flows

Volume preserving flows have the property $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}| = 1$. Each volume preserving flow is a composition of volume preserving layers where $|\frac{\partial \mathbf{y}_l}{\partial \mathbf{y}_{l-1}}| = 1$ for any $\mathbf{y}_l = f_l(\mathbf{y}_{l-1}), l = 1, 2, ..., L$. $p_X(\mathbf{x})$ can be directly computed with prior distribution such that $p_X(\mathbf{x}) = p_Z(\mathbf{z}), \mathbf{z} = f(\mathbf{x})$. The prior distribution adopted in this work is more complex than that used in general flows, and the distribution parameters are trainable variables. In this paper, the prior distribution is the mix-Gaussian distribution

$$p_Z(\mathbf{z}) = \prod_{i=1}^{d} \big[\sum_{k=1}^{K} \pi_{ik} \cdot \mathcal{N}(z_i|\mu_{ik}, \sigma_{ik}^2)\big], \tag{2}$$

where $\mathbf{z} = [z_1, ..., z_d]^\top$ and $\sum_{k=1}^{K} \pi_i = 1$. $\pi_{ik}, \mu_{ik}, \sigma_{ik}^2$ are computed with learnable parameters $\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\gamma} \in \mathbb{R}^{d \times K}$. In particular, $\pi_{ik} = \mathrm{softmax}(\boldsymbol{\alpha}_i), \mu_{ik} = \boldsymbol{\mu}_{ik}, \sigma_{ik}^2 = \exp(\boldsymbol{\gamma}_{ik})$.

For factor-out layers (shown in Sec. 3.1), $p(\mathbf{z}_1|\mathbf{y}_1)$ is modelled with gaussian distribution

$$p(\mathbf{z}_1|\mathbf{y}_1) = \mathcal{N}(\mathbf{z}_1|\boldsymbol{\mu}(\mathbf{y}_1), \exp(\boldsymbol{\gamma}(\mathbf{y}_1))), \tag{3}$$

where $\boldsymbol{\mu}(\cdot), \boldsymbol{\gamma}(\cdot)$ are all modelled with neural networks. The input dimension is the same as that of $\mathbf{y}_1$ and the output dimension is the same as that of $\mathbf{z}_1$.

From Sec. 3.1, it can be seen that the volume preserving layers can be constructed from most general flow layers with limited constraints, thus the expressive power is close to the general flow. Moreover, the complex prior with learnable parameters improves the expressive power of iVPF. In fact, compared to general flows, iVPF reaches higher bpd than most non-volume preserving ones like RealNVP [1] and Glow [4].

A.1.3 Training Volume Preserving Flows

As $p_X(\mathbf{x})$ can be directly computed, the training objective is the maximum log-likelihood. Note that we use the discrete data for training the continuous flow. In particular, the input data (images, texts, binary data, etc.) are a branch of integers such that $\mathbf{x} \in \{0, 1, ..., 2^h - 1\}^d$. To resolve this issue, we use variational dequantization technique to make the input data continuous [3] in which some noise is added to input data. Given the discrete data $\mathbf{x}$, the training objective is

$$\mathcal{L} = -\log p_X(\mathbf{x} + \mathbf{u}^\circ) + \log q(\mathbf{u}^\circ|\mathbf{x}) \quad \mathbf{u}^\circ \sim q(\mathbf{u}|\mathbf{x}) \tag{4}$$

where $\mathbf{u}^\circ \in [0,1)^d$. $q(\mathbf{u}|\mathbf{x})$ is certain distribution within $[0,1)^d$, which can be either learned distribution with parameters (e.g. flows), or pre-defined distributions.

In this paper, we simply use uniform distribution such that $q(\mathbf{u}|\mathbf{x}) = U(0,1)$[1], in which the noise is dependent with the input such that $q(\mathbf{u}|\mathbf{x}) = q(\mathbf{u})$. This implementation is simple for flow training. In fact, we may use more complex $q(\mathbf{u}|\mathbf{x})$ for better model. Moreover, for stable training, we normalise the dequantized data $\mathbf{x} + \mathbf{u}^\circ$ to $[-0.5, 0.5]$ by linear transformation before feeding the flow.

It is clear that the expected value of Eq. (4) is $\mathbb{E}_{q(\mathbf{u}|\mathbf{x})}[-\log p_X(\mathbf{x}+\mathbf{u}) + \log q(\mathbf{u}|\mathbf{x})]$. It has close relationship between Eq. (10) in the paper.

A.2. rANS Coder

rANS is an efficient entropy coding method. It is the range-based variant of Asymmetric Numeral System (ANS) [2]. Given symbol s and the probability mass function $p(s)$, s can be encoded with codelength $-\log_2 p(s)$.

rANS is very simple to implement as the encoded bits is represented by a single number. For existing code c, s can be encoded to c' as

$$c'(c,s) = \lfloor c/l_s \rfloor \cdot m + (c \mod l_s) + b_s \tag{5}$$

In the above equation, m is a large integer and usually be chosen as a power of two. Denote by $\mathrm{cdf}(s)$ as the cumulative density function such that $\mathrm{cdf}(s) = \sum_{i=1}^{s} p(s)$, we have $b_s = \lfloor \mathrm{cdf}(s-1) \cdot m \rfloor$ and $l_s = b_{s+1} - b_s$.

Given $p(s)$ and code c', the symbol can be recovered. First, we compute $b = c' \mod m$, then s can be recovered such that $b_s \le b < b_{s+1}$. This can be implemented with binary search. Then the code can be recovered with

$$c(c',s) = \lfloor c'/m \rfloor \cdot l_s + (c' \mod m) - b_s \tag{6}$$

We can easily perform bits-back coding with rANS coder. The auxiliary bits can be initialised with certain integer. Then the bits-back decoding can be performed with Eq. (6) and the code is recovered with Eq. (5). In this paper, we use rANS for coding with iVPF.

A.3. From Volume Preserving Flow to iVPF

With simple modifications, a volume preserving flow can be transformed to an iVPF. The model parameter of the iVPF is directly the parameter in the trained continuous volume preserving flow. Furthermore, the following modifications are performed:

(1) Outputs of each layer. Please refer to Sec. 3.2.1 and 3.3. The inputs and outputs of each flow layer should be k-precision quantized. While in continuous flow, quantization is not needed.

[1]Note that the uniform distribution is slightly different than used in the main paper (see Sec. 4.1), as the input data is not normalised here. In fact, they are equivalent after normalisation.

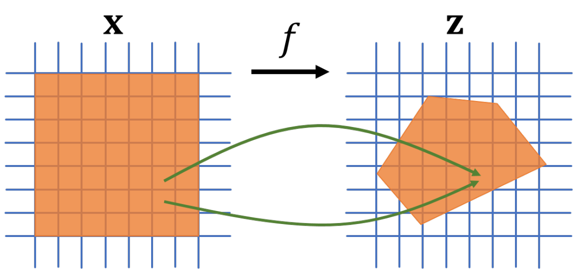

Figure 1. Illustration of how non-volume preserving flow could lead to no-bijection after discretization. The orange regions denote the domain $\mathcal{U}_0$ (in $\mathbf{x}$) and the corresponding co-domain $\mathcal{V}_0$ (in $\mathbf{z}$).

(2) Coupling layer. Please refer to Sec. 3.3.1. The results should be computed with Alg. 1. While in continuous flow, the outputs can be directly computed with Eq. (2).

(3) 1×1 convolution layer. Please refer to Sec. 3.3.2. While in continuous flow, the outputs can be directly computed with matrix multiplication of $\mathbf{W}$.

B. Proofs of Propositions

In this section, we show the proof of the two propositions in Section 3.2.2. We represent the two propositions in this section and use Fig. 1 to illustrate Proposition 2.

Proposition 1. *Let f be a smooth bijection from $\mathcal{X}$ to $\mathcal{Z}$. Assume for $\mathbf{x}_0$, it holds that $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}(\mathbf{x}_0)| > 1$. Then there exists an integer K, for any k-precision discretisation scheme where $k > K$, we have $-\log p_Z(\lfloor f(\mathbf{x}_0) \rceil)\delta > -\log p_X(\mathbf{x}_0)\delta$.*

Proof. By change of variable, we have

$$p_Z(f(\mathbf{x}_0))|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}(\mathbf{x}_0)| = p_X(\mathbf{x}_0).$$

Since $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}(\mathbf{x}_0)| > 1$, we have $p_Z(f(\mathbf{x}_0)) < p_X(\mathbf{x}_0)$. Let $\epsilon = p_X(\mathbf{x}_0) - p_Z(f(\mathbf{x}_0))$. Due to the continuity of p_Z, there exist $\delta > 0$, s.t. $p_Z(\mathbf{z}') \in (p_Z(\mathbf{z}) - \epsilon, p_Z(\mathbf{z}) + \epsilon)$, when $||\mathbf{z} - \mathbf{z}'|| < \delta$. K can be chosen st. $2^{-K-1} < \delta$ and we have

$$||f(\mathbf{x}_0) - \lfloor f(\mathbf{x}_0) \rceil|| \le 2^{-K-1} < \delta,$$

which means $p_Z(\lfloor f(\mathbf{x}_0) \rceil) < p_X(\mathbf{x}_0)$.

□

Proposition 2. *Let f be a smooth bijection from $\mathcal{X}$ to $\mathcal{Z}$. Assume for $\mathbf{x}_0$, it holds that $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}(\mathbf{x}_0)| < 1$. Then there exists an integer K, for any $k > K$, f cannot induce a bijection between the discretised domain $\bar{\mathcal{X}}$ and discretised co-domain $\bar{\mathcal{Z}}$.*

Proof. Assume $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}(\mathbf{x}_0)| < 1$, due to the continuity of the flow gradient f', there must be a neighbourhood of $\mathbf{x}_0$ denoted by $\mathcal{U}_0$, such that $|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}| < 1$ for any $\mathbf{x} \in \mathcal{U}_0$. Let $\mathcal{V}_0 = \{\mathbf{z} = f(\mathbf{x}) : \mathbf{x} \in \mathcal{U}_0\}$, we have $|\mathcal{V}_0| < |\mathcal{U}_0|$ where $|\cdot|$ denotes the volume of a space.

Consider a discretisation scheme dividing the space into bins with volume δ much smaller than $|\mathcal{U}_0|$. Since the volume of $\mathcal{V}_0$ is strictly smaller than the volume of $\mathcal{U}_0$, the number of bins in $\mathcal{V}_0$ is strictly smaller than that in $\mathcal{U}_0$. Thus a bijection could not be induced in such scenario for non-volume preserving flows. To be specific, there exists multiple $\mathbf{x}$s at different bins that are mapped to the same $\mathbf{z}$, making the lossless compression intractable. □

C. Correctness of MAT (Alg. 1)

Unless specified, we use the notation in Sec. 3.3.1.

For the encoding process, the input is $(\mathbf{x}_b, r_0)$ where $\lfloor \mathbf{x}_b \rceil = \mathbf{x}_b, r_0 \in [0, 2^C)$, the parameters are integers $m_0, m_1, ..., m_{d_b}$ where $m_0 = m_{d_b} = 2^C$, and the expected output is $(\mathbf{z}_b, r_{d_b})$. Then $(\mathbf{z}_b, r_{d_b})$ is computed as follows:

1. $\bar{\mathbf{x}}_b = 2^k \cdot \mathbf{x}_b$.
2. Given $\bar{\mathbf{x}}_b = [x_1, ..., x_{d_b}]^\top$ and r_0, compute $\mathbf{y}_b = [y_1, ..., y_{d_b}]^\top$ and $v_1, ..., v_{d_b}$ for $i = 1, ..., d_b$ such that

$$v_i = x_i \cdot m_{i-1} + r_{i-1};$$
$$y_i = \lfloor v_i / m_i \rfloor, \quad r_i = v_i \mod m_i.$$

   Then get $(\mathbf{y}_b, r_{d_b})$.
3. $\mathbf{z}_b = \mathbf{y}_b / 2^k + \lfloor \mathbf{t} \rceil$. Then get $(\mathbf{z}_b, r_{d_b})$.

For decoding process, the input is $(\mathbf{z}_b, r_{d_b})$ where $\lfloor \mathbf{z}_b \rceil = \mathbf{z}_b, r_{d_b} \in [0, 2^C)$, the parameters $m_0, m_1, ..., m_{d_b}$ are the same as that in encoding process with $m_0 = m_{d_b} = 2^C$, and the expected output is $(\mathbf{x}_b, r_0)$. Then $(\mathbf{x}_b, r_0)$ is computed as follows:

1. $\mathbf{y}_b = 2^k \cdot (\mathbf{z}_b - \lfloor \mathbf{t} \rceil)$.
2. Compute $\bar{\mathbf{x}}_b = [x_1, ..., x_{d_b}]^\top$ and $v_1 ..., v_{d_b}$ for $i = d_b, ..., 1$ such that

$$v_i = y_i \cdot m_i + r_i;$$
$$x_i = \lfloor v_i / m_{i-1} \rfloor, \quad r_{i-1} = v_i \mod m_{i-1}.$$

   Then get $(\bar{\mathbf{x}}_b, r_0)$.
3. $\mathbf{x}_b = \bar{\mathbf{x}}_b / 2^k$. Then get $(\mathbf{x}_b, r_0)$.

Then the correctness of MAT in Alg. 1 is ensured by the following theorem:

Theorem 3. *Given the above encoding and decoding process, if* $r_0 \in [0, 2^C)$*, then*
P1: $\lfloor \mathbf{z}_b \rceil = \mathbf{z}_b, r_{d_b} \in [0, 2^C)$*;*
P2: $(\mathbf{x}_b, r_0)$ *and* $(\mathbf{z}_b, r_{d_b})$ *establish a bijection.*

Proof. **P1**. Just consider the encoding process. In Step 1, $\mathbf{x}_b$ is k-precision floating points, $\bar{\mathbf{x}}_b$ are integers. In Step 2, $\mathbf{y}_b$ are integers as all elements are computed with integer space. As $r_{d_b} = v_{d_b} \mod m_{d_b}$ and $m_{d_b} = 2^C$, $r_{d_b} \in [0, 2^C)$. In Step 3, both $\mathbf{y}_b / 2^k$ and $\lfloor \mathbf{t} \rceil$ are k-precision floating points, thus $\mathbf{z}_b$ is k-precision ones with $\lfloor \mathbf{z}_b \rceil = \mathbf{z}_b$. Then the proof of **P1** is completed.

P2. Consider Step 3 in the encoding process and Step 1 in the decoding process, as $\lfloor \mathbf{z}_b \rceil = \mathbf{z}_b$, $\mathbf{z}_b$ and $\mathbf{y}_b$ establish a bijection. Consider Step 1 in the encoding process and Step 3 in the decoding process, as $\lfloor \mathbf{x}_b \rceil = \mathbf{x}_b$, $\mathbf{x}_b$ and $\bar{\mathbf{x}}_b$ establish a bijection. Then **P2** induces the bijection between $(\bar{\mathbf{x}}_b, r_0)$ and $(\mathbf{y}_b, r_{d_b})$.

Then we prove $(\bar{\mathbf{x}}_b, r_0)$ and $(\mathbf{y}_b, r_{d_b})$ establish a bijection. For ease of analysis, we rewrite the Step 2 in the decoding process such that $v_i^d = y_i \cdot m_i + r_i; x_i^d = \lfloor v_i^d / m_{i-1} \rfloor, r_{i-1}^d = v_i^d \mod m_{i-1}$. We should prove $x_i^d = x_i$ for all $i = 1, ..., d_b$ and $r_0^d = r_0$, which is done with mathematical induction.

(i) For $l = d_b$, we prove that $(x_l^d, r_{l-1}^d) = (x_l, r_{l-1})$. In fact, For the encoding process, $x_l \cdot m_{l-1} + r_{l-1} = v_l^e = y_l \cdot m_l + r_l$ and $r_l \in [0, m_l) (m_l = m_{d_b} = 2^C)$. For the decoding process, we have $v_l^d = y_l \cdot m_l + r_l = v_l = x_l \cdot m_{l-1} + r_{l-1}$. As $r_{l-1} \in [0, m_{l-1})$, it is clear that $x_l^d = \lfloor (x_l \cdot m_{l-1} + r_{l-1}) / m_{l-1} \rfloor = x_l, r_{l-1}^d = (x_l \cdot m_{l-1} + r_{l-1}) \mod m_{l-1} = r_{l-1}$.

(ii) For $1 \leq l < d_b$, if $(x_{l+1}^d, r_l^d) = (x_{l+1}, r_l)$, we prove that $(x_l^d, r_{l-1}^d) = (x_l, r_{l-1})$. For the encoding process, $x_l \cdot m_{l-1} + r_{l-1} = v_l = y_l \cdot m_l + r_l$. For the decoding process, $v_l^d = y_l \cdot m_l + r_l^d$. As $r_l^d = r_l$, we have $v_l^d = v_l$. As $r_{l-1} \in [0, m_{l-1})$, it is clear that $x_l^d = \lfloor (x_l \cdot m_{l-1} + r_{l-1}) / m_{l-1} \rfloor = x_l, r_{l-1}^d = (x_l \cdot m_{l-1} + r_{l-1}) \mod m_{l-1} = r_{l-1}$.

Note that we use a fact that $r_l \in [0, m_l), l = 1, ..., d_b$ in (i)(ii), which can be directly derived from $r_l = v_l \mod m_l$.

From (i)(ii), we have $x_l^d = x_l$ for all $l = 1, ..., d_b$ and $r_0^d = r_0$. It conveys that $(\mathbf{x}_b, r_0)$ and $(\mathbf{z}_b, r_{d_b})$ are bijections, which completes the proof of **P2**. □

D. Detailed Numerical Error Analysis of iVPF

In this section, we perform detailed numerical error analysis for iVPF. As the iVPF model mainly contains coupling layer and 1×1 convolution layer, we focus on error analysis on these layers. Overall error analysis is performed based on the composition of flow layers.

In the rest of this section, the notation is the same as that in the main paper. We first emphasise that the error between x and the quantized x at precision k, denoted by $\lfloor x \rceil$, is

$$|x - \lfloor x \rceil| \leq 2^{-k-1}.$$

Table 1. Coding efficiency of LBB and iVPF on CIFAR10. $k = 14$ is used in iVPF.

	batch size	inference time (ms)	time w/ coding (ms)	# coding
LBB [3]	64	16.2	116	188
	256	13.8	97	188
iVPF (Ours)	64	**10.9**	**11.4**	**2**
	256	**6.1**	**6.6**	**2**

D.1. MAT and Coupling Layer

For all $i = 1, 2, ..., d_b - 1$, $m_i = \text{round}(m_0 / \prod_{j=1}^{i} s_j)$; and $m_{d_b} = m_0 = \text{round}(m_0 / \prod_{j=1}^{i} s_j)$ as $\prod_{j=1}^{d_b} s_j = 1$. Then we have $m_i - 0.5 \leq m_0 / \prod_{j=1}^{i} s_j \leq m_i + 0.5$, and $m_{i-1} - 0.5 \leq m_0 / \prod_{j=1}^{i-1} s_j \leq m_{i-1} + 0.5$. Thus s_i is approximated with m_{i-1}/m_i such that

$$\frac{m_{i-1} - 0.5}{m_i + 0.5} \leq s_i \leq \frac{m_{i-1} + 0.5}{m_i - 0.5}$$

Considering s_i is fixed and $m_0 = m_{d_b} = 2^C$, it is clear that $m_i = O(2^C)$. As $\frac{m_{i-1}}{m_i} - \frac{m_{i-1} - 0.5}{m_i + 0.5} = \frac{1}{m_i} \cdot \frac{0.5(m_{i-1} + m_i)}{m_i + 0.5} = O(2^{-C})$, therefore we have $s \geq \frac{m_{i-1}}{m_i} - O(2^C)$. Similar conclusion can be arrived such that $s \leq \frac{m_{i-1}}{m_i} + O(2^C)$ and

$$|s_i - \frac{m_{i-1}}{m_i}| \leq O(2^{-C}).$$

Then we investigate the error between $y_i = \lfloor (2^k \cdot x_i \cdot m_{i-1} + r)/m_i \rfloor / 2^k$ $(0 \leq r < m_{i-1})$ and $s_i \cdot x_i$. Note that 2^k in the above equality correspond to Line 2 and 12 in Alg. 1. In fact, $y_i \leq (2^k \cdot x_i \cdot m_{i-1} + r)/(2^k \cdot m_i) < (2^k \cdot x_i \cdot m_{i-1} + m_{i-1})/(2^k \cdot m_i) = x_i \cdot m_{i-1}/m_i + m_{i-1}/(2^k \cdot m_i) = x_i \cdot m_{i-1}/m_i + O(2^{-k})$, and $y_i > [(2^k \cdot x_i \cdot m_{i-1} + r)/m_i - 1]/2^k > x_i \cdot m_{i-1}/m_i - 1/2^k = x_i \cdot m_{i-1}/m_i - O(2^{-k})$. With $|s_i - \frac{m_{i-1}}{m_i}| \leq O(2^{-C})$, the error is estimated such that

$$\begin{aligned} |y_i - s_i \cdot x_i| &\leq |y_i - \frac{m_{i-1}}{m_i} x_i| + O(2^{-C}) \\ &\leq O(2^{-k}) + O(2^{-C}) = O(2^{-k}, 2^{-C}) \end{aligned}$$

As $z_i = y_i + \lfloor t_i \rceil$ and $|\lfloor t_i \rceil - t_i| = O(2^{-k})$, we finally have

$$\begin{aligned} |z_i - (s_i x_i + t_i)| &\leq |y_i - s_i \cdot x_i| + |\lfloor t_i \rceil - t_i| \\ &= O(2^{-k}, 2^{-C}) + O(2^{-k}) = O(2^{-k}, 2^{-C}) \end{aligned}$$

which is consistent with Eq. (7) in the main paper. Thus the coupling layer in iVPF brings $O(2^{-k}, 2^{-C})$ error.

D.2. 1×1 Convolution Layer

For any $\mathbf{x}$, denote by $\tilde{z}_i$ the ith element of $\mathbf{Ux}$ and by z_i that derived in Eq. (8), it is clear that $|z_i - \tilde{z}_i| = O(2^{-k})$ as only quantization operation is involved.

For any $\mathbf{x}$, denote by $\tilde{z}_i$ the ith element of $\mathbf{\Lambda x}$ and by z_i that derived with Alg. 1, according to Sec. D.1, we have $|z_i - \tilde{z}_i| = O(2^{-k}, 2^{-C})$.

For any $\mathbf{x}$, denote by $\tilde{z}_i$ the ith element of $\mathbf{Lx}$ and by z_i that derived in Eq. (8), it is clear that $|z_i - \tilde{z}_i| = O(2^{-k})$ as only quantization operation is involved.

No numerical error is involved by performing permutation operation with the permutation matrix $\mathbf{P}$.

As the convolution layer involves sequential matrix multiplications of $\mathbf{U}, \mathbf{\Lambda}, \mathbf{L}$ and $\mathbf{P}$, the error is accumulated. Denote by $\mathbf{z} = \mathbf{PL\Lambda U}$ and $\bar{\mathbf{z}}$ the output of the corresponding iVPF layer. We have

$$|\bar{\mathbf{z}} - \mathbf{z}| = O(2^{-k}, 2^{-C}).$$

D.3. Overall Error Analysis

Denote by $\mathbf{z}$ the output of the continuous volume preserving flow model and by $\bar{\mathbf{z}}$ the output of iVPF model, then we will show that

$$|\bar{\mathbf{z}} - \mathbf{z}| = O(L2^{-k}, L2^{-C}).$$

Consider the continuous flow layer f_l and the corresponding iVPF layer $\bar{f}_l$. With the same input $\mathbf{y}_{l-1}$, we have $|\bar{f}_l(\mathbf{y}_{l-1}) - f_l(\mathbf{y}_{l-1})| = O(2^{-k}, 2^{-C})$. It usually holds that f is λ_l-Lipschitz continuous, and therefore $|f_l(\bar{\mathbf{y}}_{l-1}) - f_l(\mathbf{y}_{l-1})| \leq \lambda_l |\bar{\mathbf{y}}_l - \mathbf{y}_l|$. Thus for the volume preserving flow model with $\mathbf{y}_0 = \mathbf{x}, \mathbf{z} = \mathbf{y}_L, \mathbf{y}_l = f_l(\mathbf{y}_{l-1})$, and the corresponding iVPF model with $\bar{\mathbf{y}}_0 = \mathbf{x}, \bar{\mathbf{z}} = \bar{\mathbf{y}}_L, \bar{\mathbf{y}}_l = \bar{f}_l(\bar{\mathbf{y}}_{l-1})$, we have

$$\begin{aligned} |\bar{\mathbf{y}}_l - \mathbf{y}_l| &= |\bar{f}_l(\bar{\mathbf{y}}_{l-1}) - f_l(\mathbf{y}_{l-1})| \\ &\leq |\bar{f}_l(\bar{\mathbf{y}}_{l-1}) - f_l(\bar{\mathbf{y}}_{l-1})| + |f_l(\bar{\mathbf{y}}_{l-1}) - f_l(\mathbf{y}_{l-1})| \\ &\leq O(2^{-k}, 2^{-C}) + \lambda_l |\bar{\mathbf{y}}_{l-1} - \mathbf{y}_{l-1}| \end{aligned}$$

and therefore

$$\begin{aligned} |\bar{\mathbf{z}} - \mathbf{z}| &= \sum_{l=1}^{L} (\prod_{k=l+1}^{L} \lambda_k) O(2^{-k}, 2^{-C}) \\ &= O(L2^{-k}, L2^{-C}) \end{aligned}$$

the last equality hold as $\prod_{k=l+1}^{L} \lambda_k$ is bounded, which is guaranteed by the volume-preserving property.

From the above formula, if L is limited and 2^{-k} and 2^{-C} are relatively small, the latents generated with the iVPF model is able to approximate the true distribution.

E. More Experiments on Coding Efficiency

The experiment is conducted on PyTorch framework with Tesla P100 GPU and Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.6GHz GPU. We adapt the ANS decoding and encoding code from LBB [3], which is very time-efficient. The coding time is shown in Table 1. It is clear that the proposed iVPF method is much faster than LBB. Moreover, as LBB involves a large number of coding schemes, ANS coding takes most of the time, while model inference time takes the main time in iVPF.

References

[1] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 1

[2] Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013. 2

[3] Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3879–3888, 2019. 1, 4, 5

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