VFlow: More Expressive Generative Flows with Variational Data Augmentation
Supplementary Material

A. Verification of Assumption A1 and A2

A1 For all $p(\mathbf{x}; \boldsymbol{\theta}_{D_X}) \in \mathcal{P}_{D_X}$ and $D_Z > 0$, there exists $p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}_{D_X+D_Z}) \in \mathcal{P}_{D_X+D_Z}$, such that for all $\mathbf{x}$ and $\mathbf{z}$,

$$p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}_{D_X+D_Z}) = p(\mathbf{x}; \boldsymbol{\theta}_{D_X}) p_{\boldsymbol{\epsilon}}(\mathbf{z}).$$

A2 For all $D_Z > 0$, there exists $q(\mathbf{z}|\mathbf{x}; \boldsymbol{\phi}) \in \mathcal{Q}_{D_Z}$, such that for all $\mathbf{x}$ and $\mathbf{z}$,

$$q(\mathbf{z}|\mathbf{x}; \boldsymbol{\phi}) = p_{\boldsymbol{\epsilon}}(\mathbf{z}).$$

Let $\mathbf{x}, \mathbf{z}$ be row vectors, and $\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}$ be the horizontal concatenation of $\mathbf{x}$ and $\mathbf{z}$. We first show that the following conditions are sufficient for Assumption A1 and A2.

B1 For all $\boldsymbol{\theta}_{D_X} \in \Theta_{D_X}$ and $D_Z > 0$, there exists $\boldsymbol{\theta}_{D_X+D_Z} \in \Theta_{D_X+D_Z}$, such that for all l, $\mathbf{x}$ and $\mathbf{z}$,

$$\mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}; \boldsymbol{\theta}_{D_X+D_Z}) = \begin{bmatrix}\mathbf{f}_l(\mathbf{x}; \boldsymbol{\theta}_{D_X}) & \mathbf{z}\end{bmatrix}.$$

B2 For all $D_Z > 0$, there exists $\boldsymbol{\phi} \in \Phi_{D_Z}$, such that for all l, $\mathbf{x}$ and $\mathbf{z}$,

$$\mathbf{g}_l(\boldsymbol{\epsilon}_q; \mathbf{x}, \boldsymbol{\phi}) = \boldsymbol{\epsilon}_q.$$

Proof. Under condition B1,

$$\begin{aligned}
&\mathbf{f}(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}) \\
=&\mathbf{f}_1(\ldots(\mathbf{f}_L(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}))) \\
=&\mathbf{f}_1(\ldots(\mathbf{f}_{L-1}(\begin{bmatrix}\mathbf{f}_L(\mathbf{x}) & \mathbf{z}\end{bmatrix}))) \\
=&\mathbf{f}_1(\ldots(\mathbf{f}_{L-2}(\begin{bmatrix}\mathbf{f}_{L-1}(\mathbf{f}_L(\mathbf{x})) & \mathbf{z}\end{bmatrix}))) = \ldots \\
=&\begin{bmatrix}\mathbf{f}_1(\ldots(\mathbf{f}_L(\mathbf{x}))) & \mathbf{z}\end{bmatrix}.
\end{aligned}$$

Then,

$$\begin{aligned}
p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}_{D_X+D_Z}) &= p_{\boldsymbol{\epsilon}}(\mathbf{f}(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}; \boldsymbol{\theta}_{D_X+D_Z})) \left| \frac{\partial \mathbf{f}(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}; \boldsymbol{\theta}_{D_X+D_Z})}{\partial \begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}} \right| \\
&= p_{\boldsymbol{\epsilon}}(\begin{bmatrix}\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{D_X}) & \mathbf{z}\end{bmatrix}) \left| \frac{\partial \begin{bmatrix}\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{D_X}) & \mathbf{z}\end{bmatrix}}{\partial \begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}} \right| \\
&= p_{\boldsymbol{\epsilon}}(\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{D_X})) p_{\boldsymbol{\epsilon}}(\mathbf{z}) \left| \begin{bmatrix} \frac{\partial \mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{D_X})}{\partial \mathbf{x}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \right| \\
&= p_{\boldsymbol{\epsilon}}(\mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{D_X})) \left| \frac{\partial \mathbf{f}(\mathbf{x}; \boldsymbol{\theta}_{D_X})}{\partial \mathbf{x}} \right| p_{\boldsymbol{\epsilon}}(\mathbf{z}) \\
&= p(\mathbf{x}; \boldsymbol{\theta}_{D_X}) p_{\boldsymbol{\epsilon}}(\mathbf{z}).
\end{aligned}$$

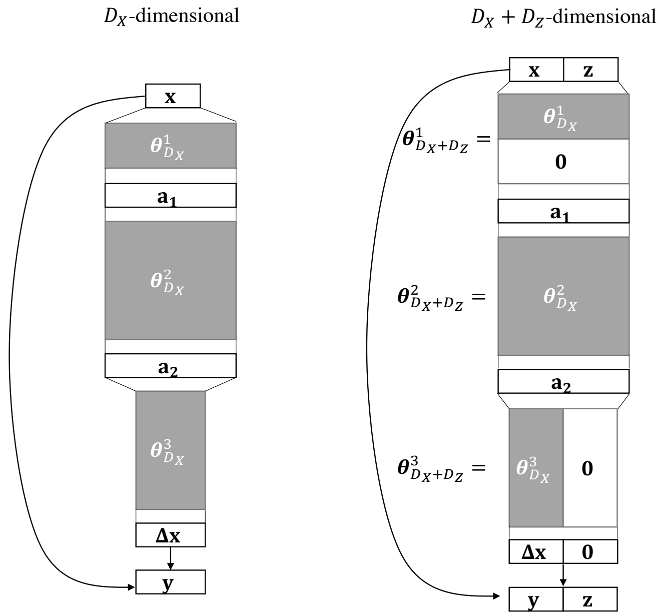

Figure 7. Constructing $\mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}; \boldsymbol{\theta}_{D_X+D_Z})$ based on $\mathbf{f}_l(\mathbf{x}; \boldsymbol{\theta}_{D_X})$.

Similarly, under condition B2,

$$\mathbf{g}(\boldsymbol{\epsilon}_q; \mathbf{x}, \boldsymbol{\phi}) = \mathbf{g}_1(\dots(\mathbf{g}_L(\boldsymbol{\epsilon}_q))) = \boldsymbol{\epsilon}_q.$$

So

$$q(\mathbf{z}; \mathbf{x}, \boldsymbol{\phi}) = q(\mathbf{g}(\boldsymbol{\epsilon}_q; \mathbf{x}, \boldsymbol{\phi})|\mathbf{x}; \boldsymbol{\phi}) = p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}_q) / \left|\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}_q}\right| = p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}_q) / |\mathbf{I}| = p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon}_q).$$

□

Therefore, we only need to verify condition B1 and B2 separately for each transformation step . For Glow (Kingma & Dhariwal, 2018) and Residual Flow (Chen et al., 2019), the transformations to verify includes affine coupling layer (Dinh et al., 2017), invertible 1×1 convolution (Kingma & Dhariwal, 2018), and invertible residual blocks (Behrmann et al., 2019). In this section we only verify condition B1 and B2 for fully-connected transformations, but they readily generalize to convolutional transformations.

A.1. Invertible Residual Blocks

An invertible residual block (Behrmann et al., 2019) $\mathbf{f}_l(\mathbf{x}; \boldsymbol{\theta}_{D_X})$ for D_X-dimensional input $\mathbf{x}$ is defined as

$$\mathbf{a}_1 = \mathbf{x}\boldsymbol{\theta}^{(1)}_{D_X}, \quad \mathbf{a}_2 = \mathbf{n}(\mathbf{a}_1; \boldsymbol{\theta}^{(2)}_{D_X}), \quad \Delta_{\mathbf{x}} = \mathbf{a}_2\boldsymbol{\theta}^{(3)}_{D_X}, \quad \mathbf{f}_l(\mathbf{x}; \boldsymbol{\theta}_{D_X}) = \mathbf{y} = \mathbf{x} + \Delta_{\mathbf{x}},$$

where we explicitly write the first and last linear layer, and leave all the internal hidden layers as $\mathbf{n}(\mathbf{a}_1; \boldsymbol{\theta}^{(2)}_{D_X})$. We construct a $D_X + D_Z$-dimensional invertible residual block $\mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}; \boldsymbol{\theta}_{D_X+D_Z})$ as

$$\begin{aligned}
&\mathbf{a}_1 = \begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}\boldsymbol{\theta}^{(1)}_{D_X+D_Z}, \quad \mathbf{a}_2 = \mathbf{n}(\mathbf{a}_1; \boldsymbol{\theta}^{(2)}_{D_X+D_Z}), \quad \begin{bmatrix}\Delta_{\mathbf{x}} & \mathbf{0}\end{bmatrix} = \mathbf{a}_2\boldsymbol{\theta}^{(3)}_{D_X+D_Z}, \\
&\mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}; \boldsymbol{\theta}_{D_X+D_Z}) = \begin{bmatrix}\mathbf{x} + \Delta_{\mathbf{x}} & \mathbf{z} + \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{f}_l(\mathbf{x}; \boldsymbol{\theta}_{D_X}) & \mathbf{z}\end{bmatrix},
\end{aligned}$$

satisfying condition B1, where

$$\boldsymbol{\theta}^{(1)}_{D_X+D_Z} = \begin{bmatrix}\boldsymbol{\theta}^{(1)}_{D_X} \\ \mathbf{0}\end{bmatrix}, \quad \boldsymbol{\theta}^{(2)}_{D_X+D_Z} = \boldsymbol{\theta}^{(2)}_{D_X}, \quad \boldsymbol{\theta}^{(3)}_{D_X+D_Z} = \begin{bmatrix}\boldsymbol{\theta}^{(3)}_{D_X} & \mathbf{0}\end{bmatrix}.$$

This construction is demonstrated by Fig. 7. Intuitively, due to the residual structure, we only need to output $\mathbf{0}$ for all the $\mathbf{z}$ dimensions. Similarly, condition B2 can be satisfied by taking $\boldsymbol{\theta}^{(3)}_{D_Z} = \mathbf{0}$, so that all the residuals are zero and the network outputs identity.

A.2. Affine Coupling Layer

An affine coupling layer (Dinh et al., 2017) $\mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X})$ for D_X-dimensional input $\mathbf{x}$ is defined as

$$\mathbf{x}_1,\mathbf{x}_2=\text{split}(\mathbf{x}),\quad \mathbf{y}_1=\mathbf{x}_1,\quad \mathbf{y}_2=\boldsymbol{\mu}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X})+\exp(\mathbf{s}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X}))\circ\mathbf{x}_2,\quad \mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X})=\text{concat}(\mathbf{y}_1,\mathbf{y}_2).$$

The case of affine coupling layer is almost identical to the invertible residual block, because both transformations have residual structures. This can be seen by noticing when $\boldsymbol{\mu}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X})=\mathbf{s}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X})=\mathbf{0}$, $\mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X})=\mathbf{x}$. We explicitly write out the first and last linear layers of $\boldsymbol{\mu}(\cdot)$ and $\mathbf{s}(\cdot)$:

$$\begin{aligned}
&\mathbf{x}_1,\mathbf{x}_2=\text{split}(\mathbf{x}),\quad \mathbf{y}_1=\mathbf{x}_1,\\
&\mathbf{a}_1=\mathbf{x}_1\boldsymbol{\theta}_{D_X}^{(a1)},\quad \mathbf{a}_2=\boldsymbol{\mu}'(\mathbf{a}_1;\boldsymbol{\theta}_{D_X}^{(a2)}),\quad \mathbf{a}_3=\mathbf{a}_2\boldsymbol{\theta}_{D_X}^{(a3)},\\
&\mathbf{b}_1=\mathbf{x}_1\boldsymbol{\theta}_{D_X}^{(b1)},\quad \mathbf{b}_2=\mathbf{s}'(\mathbf{b}_1;\boldsymbol{\theta}_{D_X}^{(b2)}),\quad \mathbf{b}_3=\mathbf{b}_2\boldsymbol{\theta}_{D_X}^{(b3)},\\
&\mathbf{y}_2=\mathbf{a}_3+\mathbf{b}_3\circ\mathbf{x}_2,\quad \mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X})=\text{concat}(\mathbf{y}_1,\mathbf{y}_2).
\end{aligned}$$

A D_X+D_Z-dimensional affine coupling layer has the form

$$\begin{aligned}
&\begin{bmatrix}\mathbf{x}_1 & \mathbf{z}_1\end{bmatrix},\begin{bmatrix}\mathbf{x}_2 & \mathbf{z}_2\end{bmatrix}=\text{split}(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix}),\quad \mathbf{y}_1=\mathbf{x}_1,\\
&\mathbf{a}_1=\begin{bmatrix}\mathbf{x}_1 & \mathbf{z}_1\end{bmatrix}\boldsymbol{\theta}_{D_X+D_Z}^{(a1)},\quad \mathbf{a}_2=\boldsymbol{\mu}'(\mathbf{a}_1;\boldsymbol{\theta}_{D_X+D_Z}^{(a2)}),\quad \begin{bmatrix}\mathbf{a}_3 & \mathbf{u}_3\end{bmatrix}=\mathbf{a}_2\boldsymbol{\theta}_{D_X+D_Z}^{(a3)},\\
&\mathbf{b}_1=\begin{bmatrix}\mathbf{x}_1 & \mathbf{z}_1\end{bmatrix}\boldsymbol{\theta}_{D_X+D_Z}^{(b1)},\quad \mathbf{b}_2=\mathbf{s}'(\mathbf{b}_1;\boldsymbol{\theta}_{D_X+D_Z}^{(b2)}),\quad \begin{bmatrix}\mathbf{b}_3 & \mathbf{w}_3\end{bmatrix}=\mathbf{b}_2\boldsymbol{\theta}_{D_X+D_Z}^{(b3)},\\
&\mathbf{y}_2=\begin{bmatrix}\mathbf{a}_3+\mathbf{b}_3\circ\mathbf{x}_2 & \mathbf{u}_3+\mathbf{w}_3\circ\mathbf{z}_2\end{bmatrix},\quad \mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix};\boldsymbol{\theta}_{D_X+D_Z})=\text{concat}(\begin{bmatrix}\mathbf{y}_1 & \mathbf{z}_1\end{bmatrix},\mathbf{y}_2),
\end{aligned}$$

We want the networks $\boldsymbol{\mu}(\cdot;\boldsymbol{\theta}_{D_X+D_Z})$ and $\mathbf{s}(\cdot;\boldsymbol{\theta}_{D_X+D_Z})$ to ignore the $\mathbf{z}_1$ part from the input, and output zero for $\mathbf{u}_3$, $\mathbf{w}_3$, so that

$$\begin{aligned}
&\begin{bmatrix}\mathbf{a}3 & \mathbf{u}_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{\mu}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X}) & \mathbf{0}\end{bmatrix},\quad \begin{bmatrix}\mathbf{b}3 & \mathbf{w}_3\end{bmatrix}=\begin{bmatrix}\mathbf{s}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X}) & \mathbf{0}\end{bmatrix}\\
&\mathbf{y}_2=\begin{bmatrix}\boldsymbol{\mu}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X})+\exp(\mathbf{s}(\mathbf{x}_1;\boldsymbol{\theta}_{D_X}))\circ\mathbf{x}_2 & \mathbf{z}_2\end{bmatrix},\quad \mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix};\boldsymbol{\theta}_{D_X+D_Z})=\begin{bmatrix}\mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X}) & \mathbf{z}\end{bmatrix},
\end{aligned}$$

so condition B1 is satisfied. We can easily achieve this by setting

$$\begin{aligned}
&\boldsymbol{\theta}_{D_X+D_Z}^{(a1)}=\begin{bmatrix}\boldsymbol{\theta}_{D_X}^{(a1)}\\ \mathbf{0}\end{bmatrix},\quad \boldsymbol{\theta}_{D_X+D_Z}^{(a2)}=\boldsymbol{\theta}_{D_X}^{(a2)},\quad \boldsymbol{\theta}_{D_X+D_Z}^{(a3)}=\begin{bmatrix}\boldsymbol{\theta}_{D_X}^{(a3)} & \mathbf{0}\end{bmatrix}\\
&\boldsymbol{\theta}_{D_X+D_Z}^{(b1)}=\begin{bmatrix}\boldsymbol{\theta}_{D_X}^{(b1)}\\ \mathbf{0}\end{bmatrix},\quad \boldsymbol{\theta}_{D_X+D_Z}^{(b2)}=\boldsymbol{\theta}_{D_X}^{(b2)},\quad \boldsymbol{\theta}_{D_X+D_Z}^{(b3)}=\begin{bmatrix}\boldsymbol{\theta}_{D_X}^{(b3)} & \mathbf{0}\end{bmatrix}.
\end{aligned}$$

Similarly, condition B2 can be satistied by setting $\boldsymbol{\theta}_{D_Z}^{(a3)}=\boldsymbol{\theta}_{D_Z}^{(b3)}=\mathbf{0}$.

A.3. Invertible 1×1 Convolution

For fully-connected cases, invertible 1×1 convolution (Kingma & Dhariwal, 2018) degenerates to a regular matrix multiplication

$$\mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X})=\mathbf{x}\boldsymbol{\theta}_{D_X},$$

where $\boldsymbol{\theta}_{D_X}$ is a non-singular matrix. We construct $\mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X+D_Z})$ such that

$$\boldsymbol{\theta}_{D_X+D_Z}=\begin{bmatrix}\boldsymbol{\theta}_{D_X} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}.$$

Clearly, $\boldsymbol{\theta}_{D_X+D_Z}$ is also non-singular, and $\mathbf{f}_l(\begin{bmatrix}\mathbf{x} & \mathbf{z}\end{bmatrix};\boldsymbol{\theta}_{D_X+D_Z})=\begin{bmatrix}\mathbf{f}_l(\mathbf{x};\boldsymbol{\theta}_{D_X}) & \mathbf{z}\end{bmatrix}$. On the other hand, condition B2 can be satisfied by setting $\boldsymbol{\theta}_{D_Z}=\mathbf{I}$.

Table 4. Model architecture for improving existing models experiment and parameter efficiency experiment.

Model	3-channel Flow++	6-channel VFlow	6-channel VFlow	6-channel VFlow
Parameters	31.4M	37.8M	16.5M	11.9M
bpd	3.08	2.98	3.03	3.08
Architecture for $p(\mathbf{x}, \mathbf{z})$: direction $(\mathbf{x}, \mathbf{z}) \to \boldsymbol{\epsilon}$				
32×32	$f_{\text{checker}}(10, 96, 32) \times 4$	$f_{\text{checker}}(10, 96, 32) \times 2$ $f_{\text{channel}}(10, 96, 32) \times 2$	$f_{\text{checker}}(10, 64, 16) \times 2$ $f_{\text{channel}}(10, 64, 16) \times 2$	$f_{\text{checker}}(10, 56, 10) \times 2$ $f_{\text{channel}}(10, 56, 10) \times 2$
	SpaceToDepth	SpaceToDepth	SpaceToDepth	SpaceToDepth
16×16	$f_{\text{channel}}(10, 96, 32) \times 2$ $f_{\text{checker}}(10, 96, 32) \times 3$	$f_{\text{checker}}(10, 96, 32) \times 2$ $f_{\text{channel}}(10, 96, 32) \times 3$	$f_{\text{checker}}(10, 64, 16) \times 2$ $f_{\text{channel}}(10, 64, 16) \times 3$	$f_{\text{checker}}(10, 56, 10) \times 2$ $f_{\text{channel}}(10, 56, 10) \times 3$
Architecture for $q(\mathbf{z}\|\mathbf{x})$: direction $\boldsymbol{\epsilon}_q \to \mathbf{z}$				
32×32	N/A	$g_{\text{checker}}(3, 96, 32) \times 4$ Sigmoid	$g_{\text{checker}}(3, 64, 16) \times 4$ Sigmoid	$g_{\text{checker}}(3, 56, 10) \times 4$ Sigmoid
Architecture for $r(\mathbf{u}\|\mathbf{x})$: direction $\boldsymbol{\epsilon}_r \to \mathbf{u}$				
32×32	$g_{\text{checker}}(2, 96, 32) \times 4$ Sigmoid	$g_{\text{checker}}(2, 96, 32) \times 4$ Sigmoid	$g_{\text{checker}}(2, 64, 16) \times 4$ Sigmoid	$f_{\text{checker}}(2, 56, 10) \times 4$ Sigmoid

B. Model Architecture

Our model architecture is directly taken from Flow++ (Ho et al., 2019), with some minor changes to make best use of the increased dimensionality.

Flow++ has three types of invertible transformation steps, activation normalization **ActNorm** (Kingma & Dhariwal, 2018), invertible 1×1 convolution **Pointwise** (Kingma & Dhariwal, 2018) and mixture-of-logistic attention coupling layer **MixLogisticAttnCoupling** (Ho et al., 2019). Each coupling layers is controlled by the number of convolution-attention hidden layers B, number of filters D_H, and number of logistic mixture components K, as mentioned in Sec. 6. There are two types of input splits for coupling layer, where **ChannelSplit** partitions input by channel, and **CheckerboardSplit** partitions input by space. Squeezing operation **SpaceToDepth** (Dinh et al., 2017) is adopted for multiscale modeling. Conditional distributions, including augmented data distribution $q(\mathbf{z}|\mathbf{x}+\mathbf{u})$ and dequantization distribution $r(\mathbf{u}|\mathbf{x})$, are implemented by adding a transformed version of $\mathbf{x}$ to the input of every coupling layer. Further denoting **TupleFlip** as flipping the two split inputs, **Inverse**$(\cdot)$ as the inverse transformation, and **MixLogisticCoupling** as MixLogisticAttnCoupling without attention, Flow++ consists the following building blocks:

$$\begin{aligned} f_{\text{checker}}(B, D_H, K) = &\text{CheckerboardSplit} \longrightarrow \text{ActNorm} \longrightarrow \text{Pointwise} \longrightarrow \text{MixLogisticAttnCoupling}(B, D_H, K) \\ &\longrightarrow \text{TupleFlip} \longrightarrow \text{Inverse(CheckerboardSplit)} \end{aligned}$$

$$\begin{aligned} f_{\text{channel}}(B, D_H, K) = &\text{ChannelSplit} \longrightarrow \text{ActNorm} \longrightarrow \text{Pointwise} \longrightarrow \text{MixLogisticAttnCoupling}(B, D_H, K) \\ &\longrightarrow \text{TupleFlip} \longrightarrow \text{Inverse(ChannelSplit)} \end{aligned}$$

$$\begin{aligned} g_{\text{checker}}(B, D_H, K) = &\text{CheckerboardSplit} \longrightarrow \text{ActNorm} \longrightarrow \text{Pointwise} \longrightarrow \text{MixLogisticCoupling}(B, D_H, K) \\ &\longrightarrow \text{TupleFlip} \longrightarrow \text{Inverse(CheckerboardSplit)} \end{aligned}$$

$$\begin{aligned} g_{\text{channel}}(B, D_H, K) = &\text{ChannelSplit} \longrightarrow \text{ActNorm} \longrightarrow \text{Pointwise} \longrightarrow \text{MixLogisticCoupling}(B, D_H, K) \\ &\longrightarrow \text{TupleFlip} \longrightarrow \text{Inverse(ChannelSplit)} \end{aligned}$$

We show the model architectures used in Sec. 6.1 and Sec. 6.3 in Table 4, where the architecture of VFlow is almost identical with the baseline Flow++, except we use both f_{checker} and f_{channel} for the 32×32 scale, while Flow++ uses only f_{checker}. Flow++ cannot use f_{channel} for the 32×32 scale because there are odd number (3) of channels. The model architectures under 4-million-parameter budget used in Sec. 6.2 are listed in Table 5. In this experiment, we use a special affine coupling

Table 5. Model architecture for ablation experiment under fixed parameter budget.

Model	3-channel Flow++	4-channel VFlow	6-channel VFlow
Parameters	4.02M	4.03M	4.01M
bpd	3.21	3.15	3.12
Architecture for $p(\mathbf{x},\mathbf{z})$: direction $(\mathbf{x},\mathbf{z}) \to \boldsymbol{\epsilon}$			
32×32	$f_{\text{checker}}(13, 32, 4)$ ×4	$f_{\text{affine}}(3, 32)$ ×1 $f_{\text{checker}}(11, 32, 4)$ ×2 $f_{\text{channel}}(11, 32, 4)$ ×2	$f_{\text{affine}}(3, 32)$ ×1 $f_{\text{checker}}(10, 32, 4)$ ×2 $f_{\text{channel}}(10, 32, 4)$ ×2
	SpaceToDepth	SpaceToDepth	SpaceToDepth
16×16	$f_{\text{channel}}(13, 32, 4)$ ×2 $f_{\text{checker}}(13, 32, 4)$ ×3	$f_{\text{checker}}(11, 32, 4)$ ×2 $f_{\text{channel}}(11, 32, 4)$ ×3	$f_{\text{checker}}(10, 32, 4)$ ×2 $f_{\text{channel}}(10, 32, 4)$ ×3
Architecture for $q(\mathbf{z}\|\mathbf{x})$: direction $\boldsymbol{\epsilon}_q \to \mathbf{z}$			
32×32	N/A	$g_{\text{checker}}(3, 32, 4)$ ×4 Sigmoid	$g_{\text{checker}}(3, 32, 4)$ ×4 Sigmoid
Architecture for $r(\mathbf{u}\|\mathbf{x})$: direction $\boldsymbol{\epsilon}_r \to \mathbf{u}$			
32×32	$f_{\text{checker}}(2, 32, 4)$ ×4 Sigmoid	$f_{\text{checker}}(2, 32, 4)$ ×4 Sigmoid	$f_{\text{checker}}(2, 32, 4)$ ×4 Sigmoid

layer to mix $\mathbf{z}$ and $\mathbf{x}$ forcibly:

$$\mathbf{y}_1 = \mathbf{z}, \quad \mathbf{y}_2 = \boldsymbol{\mu}(\mathbf{z}) + \exp(\mathbf{s}(\mathbf{z})) \circ \mathbf{x}, \quad f_{\text{affine}} = \text{concat}(\mathbf{y}_1, \mathbf{y}_2)$$

where $\boldsymbol{\mu}$ and $\mathbf{s}$ are $\mathbb{R}^{D_Z} \to \mathbb{R}^{D_X}$ functions. We empirically find that adding this special affine coupling layer accelerates the convergence for small networks. The building block with this affine coupling layer with B hidden layers and D_H hidden units is denoted as $f_{\text{affine}}(B, D_H)$ in Table 5.

Fixing Gradient Explosion

In our experiments, we find that the implementation of mixture-of-logistic attention coupling layer (Ho et al., 2019) sometimes produces huge gradients, leading the training to diverge. To see this, note that the mixture-of-logistic attention coupling layer for a given input $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ and the output $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ is defined by:

$$\text{MixLogCDF}(x; \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{s}) = \sum_{i=1}^{K} \pi_i \sigma((x - \mu_i) \cdot \exp(-s_i)), \quad \text{where} \quad \sum_{i=1}^{K} \pi_i = 1$$

$$\mathbf{y}_1 = \mathbf{x}_1, \quad \mathbf{y}_2 = \sigma^{-1}\left(\text{MixLogCDF}(\mathbf{x}_2; \boldsymbol{\pi}_\theta(\mathbf{x}_1), \mu_\theta(\mathbf{x}_1), \mathbf{s}_\theta(\mathbf{x}_1))\right) \circ \exp(\mathbf{a}_\theta(\mathbf{x}_1)) + \mathbf{b}_\theta(\mathbf{x}_1),$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. However, the inverse sigmoid may cause gradient explosion. For example, if $x = 1 - 10^{-N}$, then $\left(\sigma^{-1}(x)\right)' = \frac{1}{x(1-x)} \approx 10^N$. If for each component, $x - \mu_i$ is large and s_i is small, then $(x - \mu_i) \cdot \exp(-s_i)$ is large, and $\text{MixLogCDF}(\mathbf{x}_2; \boldsymbol{\pi}_\theta(\mathbf{x}_1), \mu_\theta(\mathbf{x}_1), \mathbf{s}_\theta(\mathbf{x}_1))$ will be close to 1, leading to gradient explosion of the inverse sigmoid function. For example, if $\pi_i = 1$, $x - \mu_i = 4$ and $s_i = -1$, we have $\text{MixLogCDF} = \sigma((x - \mu_i) \cdot \exp(-s_i)) \approx 1 - 2 \cdot 10^{-5}$ and then the gradient can be very large. We fix this issue by scaling the input of the inverse sigmoid function to $[0.05, 0.95]$:

$$\mathbf{y}_2 = \sigma^{-1}(0.05 + 0.9 * \text{MixLogCDF}(\mathbf{x}_2; \boldsymbol{\pi}_\theta(\mathbf{x}_1), \boldsymbol{\mu}_\theta(\mathbf{x}_1), \mathbf{s}_\theta(\mathbf{x}_1))) \circ \exp(\mathbf{a}_\theta(\mathbf{x}_1)) + \mathbf{b}_\theta(\mathbf{x}_1).$$

C. Extra Experiments

We further study whether it is better to put more parameters on $p(\mathbf{x}, \mathbf{z})$ or $q(\mathbf{z}|\mathbf{x})$. Under a fixed 4 million total parameter budget, we vary the parameter allocation between $p(\mathbf{x}, \mathbf{z})$ or $q(\mathbf{z}|\mathbf{x})$, and list the corresponding result and model architecture in Table 6. The result implies that it is better to put most parameters on $p(\mathbf{x}, \mathbf{z})$, supporting our claim in Sec. 4 that the variational distribution of VFlow is not necessarily as complicated as those in VAEs.

Table 6. Parameter allocation between $p(\mathbf{x}, \mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$.

	6-channel VFlow	6-channel VFlow	6-channel VFlow
Total parameters	4.00M	4.05M	4.01M
$q(\mathbf{z}\|\mathbf{x})$ parameters	0.83M	0.64M	0.36M
bpd	3.14	3.13	3.12
Architecture for $p(\mathbf{x}, \mathbf{z})$: direction $(\mathbf{x}, \mathbf{z}) \rightarrow \boldsymbol{\epsilon}$			
32×32	$f_{\text{affine}}(3, 32)\ \times 1$ $f_{\text{checker}}(8, 32, 4)\ \times 2$ $f_{\text{channel}}(8, 32, 4)\ \times 2$	$f_{\text{affine}}(3, 32)\ \times 1$ $f_{\text{checker}}(9, 32, 4)\ \times 2$ $f_{\text{channel}}(9, 32, 4)\ \times 2$	$f_{\text{affine}}(3, 32)\ \times 1$ $f_{\text{checker}}(10, 32, 4)\ \times 2$ $f_{\text{channel}}(10, 32, 4)\ \times 2$
	SpaceToDepth	SpaceToDepth	SpaceToDepth
16×16	$f_{\text{checker}}(8, 32, 4)\ \times 2$ $f_{\text{channel}}(8, 32, 4)\ \times 3$	$f_{\text{checker}}(9, 32, 4)\ \times 2$ $f_{\text{channel}}(9, 32, 4)\ \times 3$	$f_{\text{checker}}(10, 32, 4)\ \times 2$ $f_{\text{channel}}(10, 32, 4)\ \times 3$
Architecture for $q(\mathbf{z}\|\mathbf{x})$: direction $\boldsymbol{\epsilon}_q \rightarrow \mathbf{z}$			
32×32	$g_{\text{checker}}(8, 32, 4)\ \times 4$ Sigmoid	$g_{\text{checker}}(6, 32, 4)\ \times 4$ Sigmoid	$g_{\text{checker}}(3, 32, 4)\ \times 4$ Sigmoid
Architecture for $r(\mathbf{u}\|\mathbf{x})$: direction $\boldsymbol{\epsilon}_r \rightarrow \mathbf{u}$			
32×32	$f_{\text{checker}}(2, 32, 4)\ \times 4$ Sigmoid	$f_{\text{checker}}(2, 32, 4)\ \times 4$ Sigmoid	$f_{\text{checker}}(2, 32, 4)\ \times 4$ Sigmoid

References

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582, 2019.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Chen, T. Q., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9913–9923, 2019.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017.

Choi, H., Jang, E., and Alemi, A. A. Waic, but why? generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations Workshop*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. In *Advances in Neural Information Processing Systems*, pp. 3134–3144, 2019.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. In *Advances in Neural Information Processing Systems*, pp. 7509–7520, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

Gulrajani, I., Kumar, K., Ahmed, F., Taiga, A. A., Visin, F., Vazquez, D., and Courville, A. Pixelvae: A latent variable model for natural images. In *International Conference on Learning Representations*, 2017.

Gybenko, G. Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems*, 2(4): 303–314, 1989.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730, 2019.

Hoogeboom, E., Van Den Berg, R., and Welling, M. Emerging convolutions for generative normalizing flows. In *International Conference on Machine Learning*, pp. 2771–2780, 2019.

Huang, C.-W., Dinh, L., and Courville, A. Augmented normalizing flows: Bridging the gap between generative flows and latent variable models. *arXiv:2002.07101*, 2020.

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pp. 4743–4751, 2016.

Mhaskar, H. N. Approximation properties of a multilayered feedforward artificial neural network. *Advances in Computational Mathematics*, 1(1):61–80, 1993.

Morrow, R. and Chiu, W.-C. Variational autoencoders with normalizing flow decoders, 2020. URL `https://openreview.net/forum?id=r1eh30NFwB`.

Müller, T., McWilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural importance sampling. *ACM Transactions on Graphics (TOG)*, 38(5):145, 2019.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pp. 4723–4732, 2019.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Prenger, R., Valle, R., and Catanzaro, B. Waveglow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621. IEEE, 2019.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Song, Y., Meng, C., and Ermon, S. Mintnet: Building invertible neural networks with masked convolutions. In *Advances in Neural Information Processing Systems*, pp. 11002–11012, 2019.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114, 2019.

Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pp. 1747–1756, 2016.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4541–4550, 2019.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.