Learned Lossless Image Compression based on Bit Plane Slicing

Zhe Zhang[1] Huairui Wang[1] Zhenzhong Chen[1*] Shan Liu[2]
[1]Wuhan University [2]Tencent Media Lab
{zhe_zhang, zzchen}@whu.edu.cn

Abstract

Autoregressive Initial Bits (ArIB), a framework that combines subimage autoregression and latent variable models, has shown its advantages in lossless image compression. However, in current methods, the image splitting makes the information of latent variables being uniformly distributed in each subimage, and causes inadequate use of latent variables in addition to posterior collapse. To tackle these issues, we introduce Bit Plane Slicing (BPS), splitting images in the bit plane dimension with the considerations on different importance for latent variables. Thus, BPS provides a more effective representation by arranging subimages with decreasing importance for latent variables. To solve the problem of the increased number of dimensions caused by BPS, we further propose a dimension-tailored autoregressive model that tailors autoregression methods for each dimension based on their characteristics, efficiently capturing the dependencies in plane, space, and color dimensions. As shown in the extensive experimental results, our method demonstrates the superior compression performance with comparable inference speed, when compared to the state-of-the-art normalizing-flow-based methods. The code is at https://github.com/ZZ022/ArIB-BPS.

1. Introduction

Lossless image compression is widely applied in many domains where the preservation of high-quality images is demanded such as photography, scientific exploration, and remote sensing. The significance of these compression techniques lies in their ability to reduce image file sizes while preserving the original content, accomplished through the exploitation of inherent image correlations.

Fundamentally, Shannon's source coding theorem [30] sets a lower bound for coding length based on the image's entropy, which is determined by its inherent distribution. Consequently, many deep generative models [4], which have exhibited remarkable efficacy in distribution modeling, have found widespread application within the realm of lossless image compression. These encompass a variety of methodologies, including autoregressive methods (ARMs) [29, 37], variational autoencoders (VAEs) [19, 22, 34, 35], normalizing flows (NFs) [12, 13, 41, 42], and diffusion models [14, 17]. These diverse approaches involve trade-offs between compression performance and inference speed. Furthermore, compression performance encompasses dataset compression, which often approaches the theoretical limits optimized by neural networks, and single-image compression, which is affected by the initial bits required for bits back coding [34]. Bits back coding is utilized in VAEs, continuous NFs, and variational diffusion models. In summary, per pixel ARMs and diffusion models demonstrate impressive compression performance but necessitate impractical inference time [22, 28]. Among the remaining methods, continuous NFs offer the best dataset compression but exhibit weaker performance in single-image compression.

A potential option for achieving a better trade-off is the combined use of subimage autoregression and VAEs, a kind of latent variable model. This approach has shown competitive performance in the field of lossless image compression, particularly in its single-image compression performance when using the Autoregressive Initial Bits (ArIB) framework [28]. ArIB excludes latent variables for some subimages and uses their bits as initial bits, significantly reducing the overhead associated with initial bits for single-image compression.

Current methods [28] split the image in the space dimension. Pixels with the same coordinate in different subimages are adjacent in the original image. This splitting uniformly distributes the information of latent variables in each subimage. Thus, each subimage carries similar importance for latent variables. However, latent variables are inadequately used due to their exclusion for some subimages. In addition, there's a growing risk of posterior collapse [21] along the subimage sequence due to the growing volume of the autoregressive prior. Posterior collapse, a phenomenon where non-useful latent variables are learned, adversely im-

*Corresponding author. This work was supported by the National Natural Science Foundation of China (Grant No. 62036005) and Tencent.

pacts compression performance. Given the importance of all subimages for latent variables, current methods suffer from posterior collapse [28].

To address these issues, we propose the utilization of bit plane slicing (BPS). BPS splits images along the bit plane dimension, with a reduction in the volume of global data modality from the most significant plane (MSP) to the least significant plane (LSP) (see Figure 2). Given that latent variables are utilized for learning global data modalities [22, 28], the importance for latent variables decreases from MSP to LSP (see Figure 3). We arrange subimages with decreasing importance for latent variables. Hence, a more adequate utilization of latent variables is achieved by excluding subimages with minor importance. Moreover, we enhance autoregression by introducing the autoregression from MSP to LSP. This autoregression has a mitigated risk of posterior collapse since subimages important for latent variables are subject to a low risk of posterior collapse.

The introduction of BPS increases the number of dimensions of subimages. When combining autoregression in the plane dimension with that in the existing space and color dimensions, an efficient method is required. To solve the problem, we propose a dimension-tailored autoregressive model. Autoregressive methods are tailored based on the unique characteristics of each dimension: a non-neural distribution model for the simplest color space, a simple neural network for the space dimension, and a much more complex network for the challenging plane dimension. This model demonstrates a substantial improvement in performance, balanced with a suitable increase in complexity, as evidenced by our experimental results.

Consequently, our method outperforms the current ArIB methods in compression performance. Moreover, compared to the state-of-the-art NF-based methods, our approach demonstrates superior compression performance with comparable inference time.

Our contributions could be summarized as follows:

- We introduce bit plane slicing (BPS) for ArIB, since it splits the subimages with different importance for latent variables. By arranging subimages with decreasing importance for latent variables, BPS enables adequate use of latent variables as well as enhanced autoregression in a mitigated risk of posterior collapse.
- We propose a dimension-tailored autoregressive model. This model tailors autoregression methods for each dimension based on their different characteristics, efficiently modeling dependencies in an increased number of dimensions caused by BPS.
- With the merit of the BPS and dimension-tailored autoregressive model, our method surpasses the state-of-the-art normalizing-flow-based methods in compression performance with comparable inference time.

2. Related Work

Autoregressive Models compress images by predicting the probability of the current component based on previously decoded ones. Per-pixel ARMs [29, 37] decode images pixel by pixel, yielding impressive compression performance. However, as image resolution increases, the number of required network evaluations grows linearly, leading to impractically long decoding times. To mitigate this issue, subimage ARMs divide an image into a fixed number of subimages, significantly reducing inference time in comparison to per-pixel ARMs. Various image-splitting methods have been proposed, including spatial splitting [9], channel-wise splitting [24], unevenly channel-wise splitting [10], lossy and residual decomposition [23], autoregressive subpixel convolution [28], and low-high frequency decomposition [26]. In the neural compression community, autoregression in the bit plane dimension has been adopted for order-agnostic autoregressive diffusion models [14], progressive compression [15], and high bit-depth medical image compression [38]. These methods employ BPS for different goals. Our method leverages autoregression in the bit plane dimension for ArIB to enhance autoregression in a mitigated risk of posterior collapse.
Latent variable models utilize extracted latent variables from input images to aid in compression. L3C [22] employs a neural network to extract latent variables as auxiliary information. VAE-based methods [19, 34, 35] use posterior sampling for obtaining latent variables to improve performance. LBB [12] defines a posterior distribution for latent variables, enabling discretization for continuous NFs, which is also applied in iFlow [41]. However, posterior sampling requires bits back coding [34], which introduces significant overhead for single-image compression.
Combined methods integrate latent variable models and ARMs. They have witnessed great success in the neural lossy compression community [25] and have been explored in lossless compression recently [16, 28]. This combination enables the efficient capture of both local and global data modalities. SHVC [28] further introduces ArIB based on the combination to reduce the overhead by initial bits.

3. Method

In this section, we introduce our Autoregressive Initial Bits with Bit Plane Slicing (ArIB-BPS) method. To provide a clear overview of our method, Figure 1 illustrates the key components and workflow of ArIB-BPS.

3.1. Background

Subimage Autoregressive Models split the image x into a fixed number of subimages $\{x_1, x_2, \ldots, x_n\}$. The probability of x is factorized into the product of conditional distributions as follows

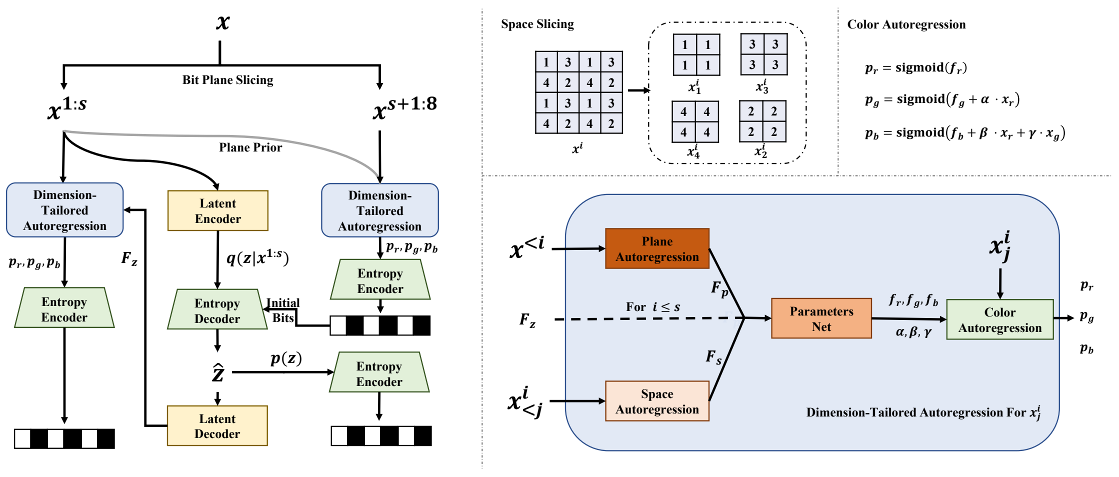

Figure 1. Overview of our method: the left subfigure illustrates the encoding process. Here, s denotes the split index. The depth of the orange indicates the complexity of the networks in the dimension-tailored autoregression. In the encoding process, the insignificant planes $x_{s+1:8}$ are first encoded conditioned on the significant planes $x_{1:s}$. Then, the posterior distribution $q(z|x_{1:s})$ is obtained from $x_{1:s}$. The latent variables $\hat{z}$ are obtained by decoding from the previously encoded bitstream using $q(z|x_{1:s})$. Next, $x_{1:s}$ is encoded, conditioned on $\hat{z}$. Finally, $\hat{z}$ is encoded with prior distribution $p(z)$. The decoding order is the reverse of the encoding process.

$$p(x) = p(x_1) \prod_{i=2}^{n} p(x_i|x_{1:i-1}) \tag{1}$$

Latent variable models extract latent variables z from the input image x to assist compression. For VAEs and continuous NFs, the approach to obtain z is sampling for a posterior distribution $q(z|x)$. In this case, bits back coding [34] is required. It encodes the image via the following steps:

Step 1: Decode z with $q(z|x)$ from initial bits.
Step 2: Encode x with $p(x|z)$.
Step 3: Encode z with $p(z)$.

Here, $p(\cdot)$ represents the prior distribution. The asymmetric numeral system (ANS) [8], an entropy coder with a reversed order between encoding and decoding, is required.

LBB [12] introduces local bits back, using $q(z|x)$ to discretize continuous NFs for lossless compression. Moreover, VAEs use neural networks to obtain $q(z|x)$. When sequentially encoding a set of images $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$, bits from previously encoded images could be used as initial bits for the current image. For VAEs, the average code length for encoding the image set is

$$\bar{C} = \frac{1}{N} \sum_{i=1}^{N} \Big(-\log p(x^{(i)}|z^{(i)}) - \log p(z^{(i)}) + \log q(z^{(i)}|x^{(i)}) \Big) - \frac{1}{N} \log q(z^{(1)}|x^{(1)}) \tag{2}$$

The first term, which is the Evidence Lower Bound (ELBO), can be optimized by neural networks. The second term can be amortized when compressing a dataset. However, when compressing a single image, the overhead $-\log q(z|x)$ is not negligible, which limits the usage of these methods for single-image compression. Nonetheless, single-image compression is required for image compression applications, and it enables parallel compression when compressing a dataset.

When combined with subimage autoregression, this overhead could be reduced via autoregressive initial bits (ArIB) [28]. ArIB partitions the subimage sequence into two subsets: $x_{1:s}$ and $x_{s+1:n}$. Latent variables are employed exclusively for $x_{1:s}$, while the bits used to compress $x_{s+1:n}$ serve as initial bits. If the number of bits provided is larger than the number of bits required, the overhead for bits back coding is eliminated.

3.2. Bit Plane Slicing for Autoregressive Initial Bits

The current method, SHVC-ArIB [28], splits the image in the space dimension. Its splitting ensures pixels with the same coordinates are adjacent in the original image. Thus, each subimage is equally important for latent variables. This splitting causes inadequate use of latent variables, as latent variables are excluded for some subimages that are important for latent variables. For SHVC-ArIB, this incurs a performance cost compared to utilizing latent variables for all subimages. Moreover, the performance of SHVC-ArIB is constrained by posterior collapse [21]. Posterior collapse refers to the collapse of the posterior to the prior, indicating that non-useful information is extracted by latent variables. A common cause of posterior collapse is highly capable decoders [21]. Specifically, a strong autoregressive prior enhances the decoder's capability, increasing the likelihood of posterior collapse. Consequently, subimages positioned at

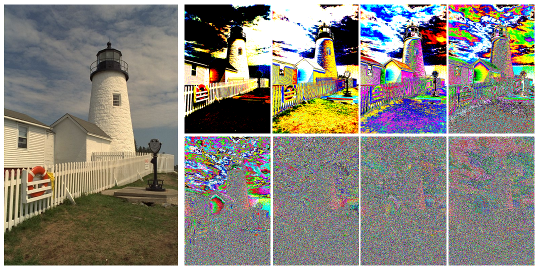

Figure 2. Visualization for each bit plane. The planes are arranged from MSP to LSP, from left-top to right-bottom. The volume of global data modality decreases from MSP to LSP.

the end of the autoregressive sequence are in a high risk of posterior collapse due to their strong autoregressive priors. In SHVC-ArIB, enhancing spatial autoregression leads to slight improvements or even a decline in performance because of posterior collapse. In this paper, we introduce bit plane slicing (BPS) to address these issues.

Bit Plane Slicing is a well-established technique in digital image processing. For a d-bit image, it partitions the entire image into d bit planes. Here, each pixel intensity is represented as x, and the l^{th} bit is represented as x^l. The value of x^l can be determined using the following equation:

$$x^l = \lfloor x/2^{d-l} \rfloor \mod 2 \tag{3}$$

In this equation, $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. The original pixel value x can be reconstructed using the equation:

$$x = \sum_{l=1}^{d} 2^{d-l} \times x^l \tag{4}$$

In this paper, we focus specifically on 8-bit images. We define the significant planes as those with smaller l values, and the insignificant planes as those with larger l values. The volume of global data modality decreases from the most significant plane (MSP) to the least significant plane (LSP), as illustrated in Figure 2. Considering latent variables are employed to learn global data modalities [22, 28], we hypothesize that the importance for latent variables similarly decreases from MSP to LSP. To test this hypothesis, we employ the latent variable model from [34] for each bit plane. Subsequently, we evaluate the Bit Per Dimension (BPD) savings and calculate the bit rate percentage attributed to latent variables within the entire subimage. The results, illustrated in Figure 3, display a consistent descending trend from MSP to LSP, with values becoming negligible in the final few insignificant planes. This observation corroborates our initial hypothesis.

Based on this finding, we employ BPS for ArIB. We partition the image into two segments using a slice index ranging in $\{1, \dots, 8\}$, denoted as s. The first segment comprises the significant planes, represented as $x^{1:s}$, while the second segment includes the insignificant planes, denoted as $x^{s+1:8}$. Latent variables z are exclusively used for the significant planes, while ARMs are employed for both segments. This approach enables a more adequate use of latent variables since $x^{1:s}$ is more important for them. Moreover, we introduce plane-by-plane autoregression from MSP to LSP to enhance autoregression. As the volume of autoregressive prior increases along the subimage sequence, the risk of posterior collapse intensifies sequentially. In this autoregression, subimages that are more important for latent variables are placed in positions where they are less likely to encounter posterior collapse. In comparison, SHVC-ArIB split the image into subimages with similar importance, meaning that subimages in positions where there's a high risk of posterior collapse are also important for latent variables. Thus, the overall risk of posterior collapse is mitigated by our method.

The overall bit rate of the image x can be expressed as follows, where R_{sig} and R_{ins} denote the bit rates associated with significant and insignificant planes, respectively:

$$\begin{aligned} R_x &= R_{sig} + R_{ins} \\ &= -\log p(x^{1:s}|z) - \log p(x^{s+1:8}|x^{1:s}) \end{aligned} \tag{5}$$

We further propose some modifications to suit our ArIB with BPS method.

Hierarchical Latent Variables are commonly utilized in VAEs to improve performance. These variables are structured in a hierarchy comprising L layers, denoted as $z = \{z_1, \dots, z_L\}$. The prior probability distribution of z is factorized as follows:

$$p(z_{1:L}) = p(z_L) \prod_{i=1}^{L-1} p(z_i|z_{i+1:L}) \tag{6}$$

There are two methods for obtaining the posterior $q(z|x^{1:s})$: top-down, which proceeds from layer L to layer 1, and bottom-up, which goes from layer 1 to layer L. The former has been shown to exhibit superior compression performance [32, 35], while the latter can leverage the bit-swap technique [19] to reduce the number of initial bits required. Given that the bit rate increases from MSP to LSP [43], insignificant planes can contribute a larger number of initial bits. Consequently, we opt for the top-down inference models, a choice that differentiates our approach from SHVC-ArIB. In this context, the posterior distribution can be expressed as follows:

$$q(z_{1:L}|x^{1:s}) = q(z_L|x^{1:s}) \prod_{i=1}^{L-1} q(z_i|z_{i+1:L}, x^{1:s}) \tag{7}$$

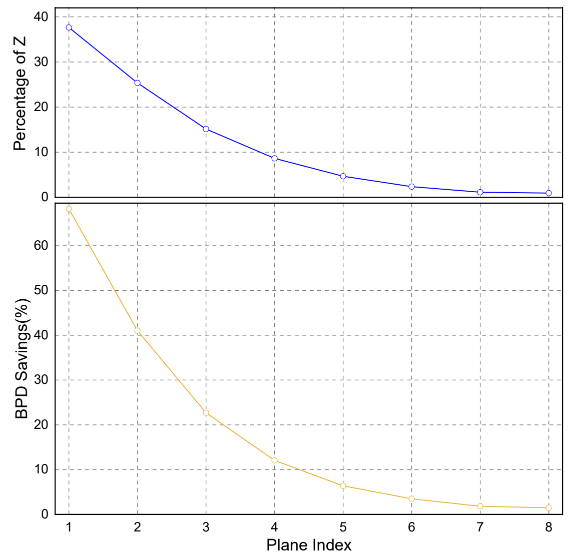

Figure 3. BPD savings achieved using latent variables and the bit rate percentage attributed to latent variables for each bit plane. The results are averaged over the CIFAR10, ImageNet32, and ImageNet64 [6] datasets. The importance for latent variables decreases from MSP to LSP.

Discretization also plays a pivotal role: a higher precision tends to enhance compression performance but also requires more initial bits. Given the variation in available initial bits across different images, we propose an image-adaptive discretization strategy.

In our method, we utilize equal mass discretization [35] for the top-down model. This ensures that all latent variables maintain equal probability mass for $p(z)$. We define the discretization precision k, which discretizes z within 2^k possible values. It is important to note that $q(z|x)$ closely approximates $p(z)$, as the Kullback-Leibler divergence between them is minimized during optimization. As a result, each element of z necessitates roughly $-\log(1/(2^k)) = k$ initial bits. Furthermore, the size of the latent variables, denoted as $|z|$, can be quantified. The bit rate R_{ins} can be determined by measuring the length of the encoded bitstream. Following these calculations, the precision k is derived using the equation:

$$k = \lfloor \frac{R_{ins}}{|z|} \rfloor \tag{8}$$

For training VAEs, the bit rate of latent variables is calculated with the condition that they are discretized to high precision. In practical scenarios, most images exhibit a sufficiently large precision value k, ensuring negligible impact on compression performance when employing discretization. Nevertheless, low precision is required in minor cases, where the discretization hurts the compression performance. To tackle this, we propose a discretized sampling technique for sampling latent variables for fine-tuning. More specifically, both the prior and posterior distributions are modeled using a univariate logistic distribution, characterized by mean parameters μ_p and μ_q, respectively. We sample ϵ from a standard logistic distribution. The calculation of a sampled latent variable $\hat{z}$ and its corresponding bit rate can be represented as follows:

$$\begin{aligned} t_l &= \lceil 2^k \cdot \text{sigmoid}(\epsilon + \mu_q - \mu_p) \rceil - 1, \\ \hat{z} &= \text{logit}\left(\frac{t_l + 0.5}{2^k}\right) + \mu_p, \\ R_{\hat{z}} &= \log(\text{sigmoid}(\text{logit}\left(\frac{t_l + 1}{2^k}\right) + \mu_p - \mu_q) \\ &\quad - \text{sigmoid}(\text{logit}\left(\frac{t_l}{2^k}\right) + \mu_p - \mu_q)) + k, \end{aligned} \tag{9}$$

where $\lceil x \rceil$ signifies the smallest integer greater than or equal to x. During back-propagation, the Straight Through Estimator [3] is utilized to manage this operation.

3.3. Dimension-Tailored Autoregressive Model

BPS results in an increased number of dimensions for autoregression, specifically, the bit plane dimension. In addition to the plane dimension, we consider autoregression in space and color dimensions, which have demonstrated their efficacy in previous studies [26, 40]. Using a uniform approach to model all dimensions is employed in previous works [14]. However, this method has certain limitations. First, these dimensions present uneven modeling challenges. Employing a one-size-fits-all network to address all dimensions directly may either compromise modeling accuracy in the more complex dimensions or lead to inefficient use of computational resources in simpler dimensions. Second, strong autoregression in the space and color dimensions can potentially lead to posterior collapse. To address these issues, we propose a dimension-tailored autoregressive model that tailors the approach for each dimension based on its specific characteristics. An overview is shown in the right subfigures of Figure 1.

Our autoregressive model operates in three stages. In the first stage, we focus on the plane dimension, which presents a significant challenge due to the extensive range of possible values that the pixels in its preceding planes, denoted as $x^{1:i-1}$, can take on, amounting to 2^{i-1} possibilities. To effectively leverage the autoregressive prior from $x^{1:i-1}$ and apply it to x^i, we employ a sophisticated plane context model, Φ_p. To capture the global data modality embedded in the preceding planes, we adopt a U-Net architecture [27]. Specifically, we use two models that employ the architecture of diffusion models [7]: one for the significant planes and the other for the insignificant planes. Time embedding is used to differentiate planes.

In the second stage, we address the space dimension. Compared to the plane dimension, this dimension is less complex, as each bit in the current bit plane can only take one of two possible values. We adopt a 4-stage checkerboard model [9, 26], which involves dividing the bit plane x^i into four sub-planes, denoted as $x_1^i, x_2^i, x_3^i, x_4^i$, as illustrated in Figure 1. For each sub-plane x_j^i, we utilize a simple network, Φ_s, to extract the spatial autoregressive prior F_s from $x_{1:j-1}^i$. Then, a parameter network Θ is used to fuse information from both the plane and space dimensions. To effectively integrate information, we employ a relatively complex network for Θ. Although the complexity of Θ does contribute to the overall computational demand of our model, the complexity remains significantly lower than that of employing the architecture of Φ_p for both dimensions, resulting in a more efficient modeling process. More specifically, Φ_s is a single ResBlock [11], and Θ combines an ECANet [39] with several wider ResBlocks.

The third stage is for the color dimension. As identified in PixelCNN++ [29], the correlations among the color channels of a pixel are typically straightforward, and deep networks are often unnecessary for their modeling. Therefore, we adopt a non-network approach for the color dimension. Specifically, we model the distribution of bits in each color channel using a Bernoulli distribution, where the probability parameter $p = P(x = 1)$ is determined autoregressively based on the bits with the same spatial position in previously decoded channels. In our method, Θ predict six parameters $\theta = \{f_r, f_g, f_b, \alpha, \beta, \gamma\}$, and model the probability for each channel as follows:

$$\begin{aligned} p_r &= \text{sigmoid}(f_r) \\ p_g &= \text{sigmoid}(f_g + \alpha \cdot x_r) \\ p_b &= \text{sigmoid}(f_b + \beta \cdot x_r + \gamma \cdot x_g) \end{aligned} \tag{10}$$

The 1st and 2nd stages of our dimension-tailored autoregressive model can be expressed as follows:

$$\begin{aligned} F_{p^i} &= \Phi_p(x^{1:i-1}) \\ F_{s_j^i} &= \Phi_s(x_{1:j-1}^i) \\ \theta_j^i &= \begin{cases} \Theta(F_{p^i}, F_{s_j^i}, F_z), & i \leq s \\ \Theta(F_{p^i}, F_{s_j^i}), & i > s \end{cases} \end{aligned} \tag{11}$$

In summary, we initially derive F_p for the entire bit plane. Subsequently, for each of the four subimages in the bit plane, we compute F_s. These are then concatenated with F_p and F_z (only for significant planes) and fed into Θ to determine the six parameters θ. Finally, we calculate the probability using non-network color autoregression as outlined in Equation 10.

Loss function of our method is expressed as:

$$\begin{aligned} l &= R_z + R_x \\ &= -\log p(z) + \log q(z|x^{1:s}) + R_x \end{aligned} \tag{12}$$

4. Experiments

4.1. Architecture and Training Details

In accordance with prior research, our experimental setup entails training three models using three benchmark datasets: CIFAR10, ImageNet32, and ImageNet64 [6]. For all models, we set the split index, s, to 4, based on Figure 3, and utilize 3 layers of hierarchical latent variables. For latent variables, we apply a factor 2 downsampling for each hierarchy of latent variables in latent encoders. The channel number for latent variables is set to 3 across all models. For Φ_p, the number of hierarchical levels in the U-Net is set to 4 across all models.

We utilize the Adam optimizer [18] with an initial learning rate of 0.0001, and we apply step decay as the learning rate schedule. For fine-tuning with discretized sampling, we employ Equation 8 with the previously trained model to obtain the precision k for each image in the training set. All experiments are conducted on a GeForce GTX 1080 Ti.

4.2. Compression Performance

We evaluate the performance of our method across four distinct datasets: CIFAR10, ImageNet32, ImageNet64, and DIV2K [1], using bits per dimension (BPD) as our primary metric. The first three datasets are of low resolution, while the last one, DIV2K, is a high-resolution dataset. Following most previous works, we exclude per-pixel autoregressive methods [29, 37] and diffusion models [14, 17] from our evaluation due to their impractical time complexity.

Our method evaluates performance in two distinct compression settings: dataset compression and single-image compression. In dataset compression, we sequentially compress 5,000 images with a fixed discretization precision of 10. For single-image compression, each image's k value is calculated using Equation 8 and clipped between 3 and 10. If the calculated k is less than 3, additional $(3-k) \times |z|$ ($|z|$ represents the size of z) bits are added to ensure successful encoding. These possible bits, along with the k value and image bits, are written to a bitstream. The single image compression performance is determined by measuring the bitstream size. Following [28, 41, 42], we evaluate the performance on the DIV2K dataset using the model trained on ImageNet64 by cropping images into 64×64 patches.

We compare our method against advanced approaches from traditional methods, learned methods without bits back coding, and learned methods with bits back coding. We present the detailed results of our experiments in Table 1. Our method demonstrates superior compression performance, particularly in single-image compression. In dataset compression, our method excels in three out of four datasets, achieving the second-best performance on the remaining dataset. In single-image compression, our method outperforms other methods in all datasets. We note

Table 1. Compression performance in BPD for four datasets, where lower values indicate better performance. The best performance is highlighted in bold. *: ImageNet32 , **: ImageNet64, $^{+}$: Flickr2k [33], $^{-}$: OpenImage [20]. 1st Group: traditional methods, 2nd Group: learned methods without bits back coding, 3rd Group: learned methods with bits back coding. Some values of SHVC and SHVC-ArIB are not available, and we list their available lower bound here. $^{\triangle}$: These values are obtained by adding dataset performance and initial bits.

	Dataset Compression				Single-Image Compression			
Method	CIFAR10	ImageNet32	ImageNet64	DIV2K	CIFAR10	ImageNet32	ImageNet64	DIV2K
PNG [5]	5.89	6.39	5.74	4.23	5.89	6.39	5.74	4.23
FLIF [31]	4.19	4.52	4.54	2.91	4.19	4.52	4.54	2.91
JPEG-XL [2]	5.74	6.39	5.89	2.79	5.74	6.39	5.89	2.79
L3C [22]	-	4.76	4.42	3.09^{-}	-	4.76	4.42	3.09^{-}
IDF [13]	3.34	4.18	3.90	-	3.34	4.18	3.90	-
IDF++ [36]	3.26	4.12	3.81	-	3.26	4.12	3.81	-
LC-FDNet [26]	3.77^{+}	4.40^{+}	4.28^{+}	2.71^{+}	3.77^{+}	4.40^{+}	4.28^{+}	2.71^{+}
Bit-Swap [19]	3.82	4.50	-	-	6.53	6.97	-	-
Hilloc [35]	3.56^{*}	4.20	3.90^{*}	-	-	-	-	-
SHVC [28]	3.16	3.98	3.68	2.57^{**}	>3.96	>4.78	>4.48	-
SHVC-ArIB [28]	>3.16	>3.98	>3.68	-	>3.16	>3.98	>3.68	-
iVPF [42]	3.20	4.03	3.75	2.68^{**}	$9.20^{\triangle}$	-	-	-
LBB [12]	3.12	**3.88**	3.70	-	$42.98^{\triangle}$	$49.84^{\triangle}$	$41.70^{\triangle}$	-
iFlow [41]	3.12	**3.88**	3.70	2.57^{**}	$37.40^{\triangle}$	$38.27^{\triangle}$	$38.12^{\triangle}$	-
ArIB-BPS (ours)	**3.06**	3.91	**3.63**	**2.55****	**3.07**	**3.92**	**3.64**	**2.59****

Table 2. Comparison with advanced and available flow-based methods. The batchsize is 64 following LBB and iFlow. We train the small model to align with the inference speed of iFlow on ImageNet64.

		Compression Performance (BPD) ↓			Inference Time (ms/sample) ↓	
dataset	Method	theoretical	dataset	single	encode	decode
CIFAR10	LBB [12]	3.116	3.118	42.978	64.94	64.94
	iFlow [41]	3.116	3.118	37.398	17.38	47.56
	ArIB-BPS	3.048	3.057	3.070	14.93	14.93
ImageNet32	LBB [12]	3.871	3.875	49.835	194.14	194.14
	iFlow [41]	3.871	3.873	38.273	74.84	119.30
	ArIB-BPS	3.904	3.911	3.918	52.96	52.96
ImageNet64	LBB [12]	3.701	3.703	41.703	95.57	95.57
	iFlow [41]	3.701	3.703	38.123	37.49	58.08
	ArIB-BPS (Small)	3.677	3.685	3.693	43.65	43.65
	ArIB-BPS	3.624	3.633	3.641	107.80	107.80

that precise values for the single-image compression performance of SHVC, as well as the compression performance of SHVC-ArIB, are not available. However, the lower bound could be obtained. Moreover, it could be inferred that our method surpasses SHVC-ArIB by an average of more than 0.10 BPD on low-resolution datasets in both compression settings. Additional details are provided in the Appendix.

To further demonstrate the effectiveness of our approach, we conducted a detailed comparison between our method and continuous normalizing-flow-based methods, which exhibit state-of-the-art performance in dataset compression. As Table 2 illustrates, ArIB-BPS outperforms iFlow in both single-image and dataset compression performance for CIFAR10 and ImageNet64, with comparable inference time. For ImageNet32, although ArIB-BPS lags slightly behind in dataset compression performance, it excels in other metrics, particularly in single-image compression. These results imply that our method achieves a better trade-off between compression performance and inference speed.

4.3. Ablation Studies

We conduct three ablation studies on the CIFAR10 dataset to show the efficiency of our dimension-tailored autoregressive model and the effectiveness of discretized sampling. We note that results in Table 3 and 4 are evaluated using models without using discretized sampling for fine-tuning. Therefore, the results of ArIB-BPS are slightly different from those in Table 2.

Table 3. Ablation for the effectiveness of autoregression on plane, space, and color dimensions. Inference time is identical for encoding and decoding. The batchsize is 64.

plane	space	color	Theoretical BPD ↓	Inference Time (ms/sample) ↓
✓	✓	✓	3.052	14.93
✓	✓		3.458	14.92
✓		✓	3.470	10.36
	✓	✓	3.522	5.83

Table 4. Compression performance and inference time of various usages of the architecture of Φ_p. The batch size is 64. P: plane. S: space. C: color. Sig: significant planes. Ins: insignificant planes. We also train two small models to further validate our design.

	Theoretical BPD ↓			Inference Time (ms/sample) ↓		
	Sig	Ins	Total	Sig	Ins	Total
P (Ours)	1.041	2.011	3.052	6.30	8.64	14.93
PS	1.052	1.993	3.045	14.46	24.97	39.42
PSC	1.043	1.971	3.015	43.83	74.97	118.80
PS (Small)	1.055	2.070	3.125	9.08	11.44	20.51
PSC (Small)	1.096	2.218	3.314	9.64	13.96	23.59

4.3.1 Dimension-Tailored Autoregressive Model

Firstly, we conduct experiments to assess the impact of autoregression in each dimension. To do this, we train and evaluate three additional models, each with autoregression removed in either the color, space, or plane dimension. In the model that does not incorporate plane autoregression, we employ the color autoregressive model from PixelCNN++ [29] to ensure compatibility. Additionally, we enhance Φ_s by employing the architecture of Φ_p, accommodating the increased complexity in the spatial dimension, which now has 256 possible values per pixel instead of just 2. In our experiments, this configuration results in improved performance. Table 3 clearly demonstrates that incorporating autoregression in each dimension consistently leads to significant improvements in compression performance. We note that our autoregression for the color dimension incurs only a negligible overhead in inference time.

Furthermore, we conduct an evaluation to assess the efficiency of our dimension-tailored design. Specifically, we train two additional models: one using the architecture of Φ_p for both the plane and space dimensions and the other using the architecture of Φ_p for all three dimensions. Subsequently, we evaluate the impact of these modifications on significant and insignificant planes separately. As depicted in Table 4, the dimension-tailored autoregressive model achieves superior performance in significant planes with lower complexity. This superiority is attributed to reduced hindrance to the latent variables, facilitated by simpler spatial autoregression and color-wise autoregression, as reflected in higher bit rates for latent variables. In the case of insignificant planes, although strengthened autoregression can lead to performance gains, it results in significantly longer inference time. To further validate our method's efficiency, we train two small models with reduced inference time. Our method demonstrates better compression performance. Considering the trade-off between inference time and compression performance, our design is more efficient.

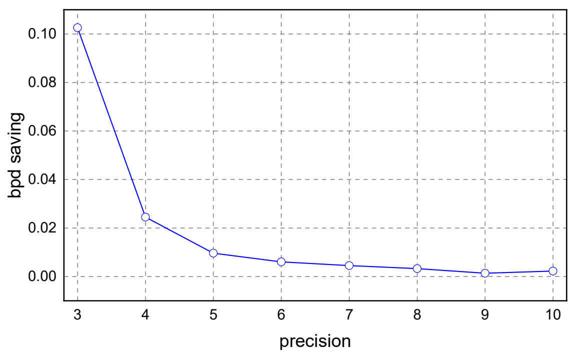

Figure 4. Average BPD savings through fine-tuning using discretized sampling across various discretization precisions.

4.3.2 Discretized Sampling

We assess the performance improvement after fine-tuning across various k values, as illustrated in Figure 4. The method improves performance across all precision values, particularly at lower precision levels. While low-precision cases form a minor segment, their optimization boosts performance in edge cases.

5. Conclusion

We have proposed ArIB-BPS, a new approach for lossless image compression. BPS enables us to achieve adequate use of latent variables as well as enhanced autoregression in a mitigated risk of posterior collapse by arranging subimages with decreasing importance for latent variables. In addition, we present a dimension-tailored autoregressive model that takes into account the different characteristics of each dimension. It efficiently captures dependencies in space, color, and plane dimensions. Experimental results demonstrate that our method achieves superior compression performance compared to the state-of-the-art normalizing-flow-based methods with comparable inference time.

The limitation of our work lies in the complexity of higher bit-depth. The complexity of our method would increase as the bit-depth increases due to the interdependency among bit planes. To extend our method for high-bit-depth images, efficient strategies such as handling multiple planes in one step should be explored in our future work.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 6

[2] Jyrki Alakuijala, Ruud Van Asseldonk, Sami Boukortt, Martin Bruse, Iulia-Maria Comșa, Moritz Firsching, Thomas Fischbacher, Evgenii Kliuchnikov, Sebastian Gomez, Robert Obryk, et al. Jpeg xl next-generation image compression architecture and coding tools. In *Applications of Digital Image Processing XLII*, pages 112–124. SPIE, 2019. 7

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 5

[4] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris George Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

[5] Thomas Boutell. Png (portable network graphics) specification version 1.0. Technical report, 1997. 7

[6] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 5, 6

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 5

[8] Jarek Duda. Asymmetric numeral systems. *arXiv preprint arXiv:0902.0271*, 2009. 3

[9] Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14771–14780, 2021. 2, 6

[10] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5718–5727, 2022. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6

[12] Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2, 3, 7

[13] Emiel Hoogeboom, Jorn Peters, Rianne Van Den Berg, and Max Welling. Integer discrete flows and lossless compression. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 7

[14] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022. 1, 2, 5, 6

[15] Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. Context-based trit-plane coding for progressive image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14348–14357, 2023. 2

[16] Ning Kang, Shanzhao Qiu, Shifeng Zhang, Zhenguo Li, and Shu-Tao Xia. Pilc: Practical image lossless compression with an end-to-end gpu oriented neural framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3739–3748, 2022. 2

[17] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in Neural Information Processing Systems*, 34:21696–21707, 2021. 1, 6

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Friso Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pages 3408–3417, 2019. 1, 2, 4, 7

[20] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2017. 7

[21] James Lucas, George Tucker, Roger B Grosse, and Mohammad Norouzi. Don't blame the elbo! a linear vae perspective on posterior collapse. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 3

[22] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10629–10638, 2019. 1, 2, 4, 7

[23] Fabian Mentzer, Luc Van Gool, and Michael Tschannen. Learning better lossless compression using lossy compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6638–6647, 2020. 2

[24] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing*, pages 3339–3343. IEEE, 2020. 2

[25] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in Neural Information Processing Systems*, 31, 2018. 2

[26] Hochang Rhee, Yeong Il Jang, Seyun Kim, and Nam Ik Cho. Lc-fdnet: Learned lossless image compression with frequency decomposition network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6033–6042, 2022. 2, 5, 6, 7

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmen-

tation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015. 5

[28] Tom Ryder, Chen Zhang, Ning Kang, and Shifeng Zhang. Split hierarchical variational compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 386–395, 2022. 1, 2, 3, 4, 6, 7

[29] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017. 1, 2, 6, 8

[30] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. 1

[31] Jon Sneyers and Pieter Wuille. Flif: Free lossless image format based on maniac compression. In *IEEE International Conference on Image Processing*, pages 66–70. IEEE, 2016. 7

[32] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in Neural Information Processing Systems*, 29, 2016. 4

[33] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 114–125, 2017. 7

[34] James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations*, 2018. 1, 2, 3, 4

[35] James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: lossless image compression with hierarchical latent variable models. In *International Conference on Learning Representations*, 2019. 1, 2, 4, 5, 7

[36] Rianne van den Berg, Alexey A Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, and Tim Salimans. Idf++: Analyzing and improving integer discrete flows for lossless compression. In *International Conference on Learning Representations*, 2020. 7

[37] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1747–1756, 2016. 1, 2, 6

[38] Kai Wang, Yuanchao Bai, Deming Zhai, Daxin Li, Junjun Jiang, and Xianming Liu. Learning lossless compression for high bit-depth medical imaging. In *2023 IEEE International Conference on Multimedia and Expo*, pages 2549–2554. IEEE, 2023. 2

[39] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11534–11542, 2020. 6

[40] Honglei Zhang, Francesco Cricri, Hamed R Tavakoli, Nannan Zou, Emre Aksu, and Miska M Hannuksela. Lossless image compression using a multi-scale progressive statistical model. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 5

[41] Shifeng Zhang, Ning Kang, Tom Ryder, and Zhenguo Li. iflow: Numerically invertible flows for efficient lossless compression via a uniform coder. *Advances in Neural Information Processing Systems*, 34:5822–5833, 2021. 1, 2, 6, 7

[42] Shifeng Zhang, Chen Zhang, Ning Kang, and Zhenguo Li. ivpf: Numerical invertible volume preserving flow for efficient lossless compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 620–629, 2021. 1, 6, 7

[43] Zhizheng Zhang, Zhibo Chen, Jianxin Lin, and Weiping Li. Learned scalable image compression with bidirectional context disentanglement network. In *IEEE International Conference on Multimedia and Expo*, pages 1438–1443. IEEE, 2019. 4