CBANet: Toward Complexity and Bitrate Adaptive Deep Image Compression Using a Single Network

Jinyang Guo, Dong Xu, Fellow, IEEE, and Guo Lu, Member, IEEE

Abstract—In this work, we propose a new deep image compression framework called Complexity and Bitrate Adaptive Network (CBANet) that aims to learn one single network to support variable bitrate coding under various computational complexity levels. In contrast to the existing state-of-the-art learning-based image compression frameworks that only consider the rate-distortion trade-off without introducing any constraint related to the computational complexity, our CBANet considers the complex rate-distortion-complexity trade-off when learning a single network to support multiple computational complexity levels and variable bitrates. Since it is a non-trivial task to solve such a rate-distortion-complexity related optimization problem, we propose a two-step approach to decouple this complex optimization task into a complexity-distortion optimization subtask and a rate-distortion optimization sub-task, and additionally propose a new network design strategy by introducing a Complexity Adaptive Module (CAM) and a Bitrate Adaptive Module (BAM) to respectively achieve the complexity-distortion and rate-distortion trade-offs. As a general approach, our network design strategy can be readily incorporated into different deep image compression methods to achieve complexity and bitrate adaptive image compression by using a single network. Comprehensive experiments on two benchmark datasets demonstrate the effectiveness of our CBANet for deep image compression. Code is released at https://github.com/JinyangGuo/CBANet-release.

Index Terms—Deep learning, image compression, variable bitrate coding, dynamic computational complexity.

I. INTRODUCTION

IMAGE compression techniques are widely used to decrease the number of bits required for storage and transmission. Although the learning-based image compression approaches [1], [2], [3] have achieved state-of-the-art image compression performance, there are still two challenges when deploying the learning-based image compression methods for real-world applications: (1) dynamic computational complexity constraint and (2) variable bitrate coding.

First, a learning-based image compression framework that can support dynamic computational complexity constraints is required because the computational complexity constraints often change when deploying the image codecs on different platforms. For example, when using a desktop with relatively rich computational resources to watch images/movies from Netflix, we may want to use a decoder with more computational cost to generate high-quality images/movies for better visual quality, while when using an outdated iPhone 4 with limited computational resources to watch images/movies, we may have to use a decoder with less computational cost to generate low-quality images/movies for smooth decoding. Besides, the allocated computational resources for one deployment platform may also change over time (e.g., when the battery status of mobile phones changes). When the battery of a mobile phone is fully charged, we may want to use more computational resources to decode images/movies for better visual quality, while when the mobile phone entering the power saving mode, we may have to use less computational resources to decode images/videos.

Second, a learning-based image compression framework that can support variable bitrate coding is also required when the bandwidth condition changes over time. When the network bandwidth is large (e.g., 5G network), we may want to use a decoder that supports a higher bitrate to generate high-quality images/movies. When the bandwidth condition is poor, we may have to use a decoder that supports a lower bitrate to generate low-quality images/movies.

However, the existing deep image compression methods like [1], [2], and [3] need to train different deep models to support different computational complexity levels and multiple bitrates. Training and storing multiple deep models for supporting the tremendous scenarios mentioned above is time-consuming and also increases the storage burden. As shown in Fig. 1(a), we are often given dynamic computational complexity constraints and changing bandwidth conditions in real-world deployment scenarios. If we train multiple learning-based codecs at different bitrates and under various computational complexity constraints, it is obvious that the required number of decoders is equal to the number of computational complexity constraints multiplied by the number of bitrates, which significantly increases the design and storage cost for multiple models.

Therefore, a new research problem emerges: How to design an image compression system to support multiple bitrates under dynamic computational complexity constraints by using a single network? To tackle this problem, we first propose a new objective function for deep image compression, which considers the rate-distortion-complexity trade-off when learning a single network to support multiple computational
Fig. 1. Comparison of (a) the existing learning-based image compression framework, which requires multiple decoders for decoding the images at different bitrates on different devices with different computational complexity levels and (b) our CBANet, which can decode the bitstreams at different bitrates on different devices with different computational complexity constraints by using one single decoder.

complexity levels and variable bitrates. However, it is a non-trivial task to solve the objective function as the three factors (i.e., the rate, the distortion, and the complexity) are closely entangled together in the newly proposed objective function, namely, changing one of these three factors will also affect the other two factors. Although the existing works [4], [5], [6] also consider the rate, distortion, and complexity factors when using a single network, these methods fail to decouple the rate and the complexity. In other words, when changing the complexity of the network, the rate will be also changed. However, these two factors may not change simultaneously in real scenarios (e.g., the battery situation may not suddenly change when the network condition changes). To decouple the rate and the complexity, we propose a simple and effective two-step approach to decouple the complex rate-distortion-complexity optimization task as two related optimization sub-tasks corresponding to the rate-distortion trade-off and the complexity-distortion trade-off, respectively. In addition, we propose a new network design strategy, in which we introduce two separate modules called the complexity adaptive module (CAM) and the bitrate adaptive module (BAM) to respectively solve the sub-tasks related to complexity-distortion and rate-distortion trade-offs. We can readily incorporate our new network design strategy with the classical image compression pipelines [2], [3], [7] and extend these works [2], [3], [7] towards complexity and bitrate adaptive deep image compression by using a single network. Considering that the decoder side is more sensitive to the computational complexity or the storage requirement, in this work, we only focus on the decoder side. As a result, our Complexity and Bitrate Adaptive Networks (CBANet) can reconstruct the images at arbitrary bitrates under various computational complexity levels without requiring the time-consuming re-training process (see Fig. 1(b)), which cannot be achieved by the existing deep image compression approaches [2], [3], [7].

Specifically, our CBANet consists of two newly introduced modules: a CAM and a BAM. The CAM consists of several parallel branches, in which each branch takes a small portion of the computational budget and the reconstructed images with different visual qualities can be readily generated by using different numbers of branches with different computational costs. To support variable bitrate coding with one single model, the BAM uses a small convolutional neural network (CNN) to project the representation at a base bitrate to the expected representation at a target bitrate for transmission. Then it will convert the representation at the target bitrate to that at the base bitrate for the decoding process of the CAM. By seamlessly integrating the proposed CAM and BAM into a single network and optimizing our CBANet with the newly proposed two-step optimization approach, we can use one deep model to support variable bitrates under various computational complexity levels.

The main contributions of this work can be summarized as follows:

- We propose the first deep image compression work that can learn a single deep model to support various bitrates under dynamic computational complexity constraints.
- We also propose a new rate-distortion-complexity optimization problem to achieve complexity and bitrate adaptive deep image compression using a single network. To solve this optimization problem, we propose a simple and effective two-step approach to decouple the complex rate-distortion-complexity optimization task as two sub-tasks related to complexity-distortion and rate-distortion trade-offs, respectively. We further propose a new network design strategy by using two modules (i.e., a CAM and a BAM) to respectively solve these two sub-tasks. Our network design strategy is a general approach and can be readily incorporated into different image compression pipelines.
- Comprehensive experiments on two benchmark datasets demonstrate the effectiveness of our CBANet for image compression.

II. RELATED WORK

A. Image Compression

In the past decades, a large number of conventional image compression standards were proposed, such as JPEG [8], JPEG 2000 [9], and BPG [10]. However, these compression standards heavily rely on hand-crafted techniques. In recent years,
several learning-based image compression methods were proposed to improve the compression performance, among which the convolutional neural network (CNN) based methods are attracting increasing attention. For example, Ballé et al. [1] proposed an end-to-end optimized image compression system by minimizing the rate-distortion trade-off [11], which was subsequently extended by additionally introducing the hyper-prior model in [2]. To achieve better bitrate estimation, the method in [3] further extended the work in [2] by introducing the context model for adaptive arithmetic coding. Building upon the work in [3], the image-dependent entropy models were additionally introduced in [12]. Recently, Cheng et al. [7] utilizes the discretized Gaussian mixture likelihoods technique to parameterize the distributions of latent codes. In addition to the aforementioned line of works, other methods were also proposed for image compression. Agustsson et al. [13] proposed the soft-to-hard quantization approaches for deep image compression. Theis et al. [14] uses the compressive autoencoders to deal with the non-differentiable problem caused by the quantization operation. Mentzer et al. [15] introduced a 3D CNN based context model to better model the entropy of the latent representation. Agustsson et al. [16] utilizes the generative adversarial networks for deep image compression. Lee et al. [17] proposed a new context-adaptive entropy model for better image compression performance. Li et al. [18] introduced the content-weighted importance map for content-aware bitrate allocation. Minnen et al. [19] splits the latent representation into different parts and processes different parts by using different entropy information. Hu et al. [20] proposed to improve the rate-distortion performance in a coarse-to-fine fashion. These methods aim to improve the deep image compression performance while ignoring the efficiency of the image compression systems. So the goals of these methods are intrinsically different with our approach. Johnston et al. [21] proposed to utilize the neural architecture search technique to search the optimal image compression model under a pre-defined computational budget. Rippel et al. [22] introduced the adaptive codeweight regularization to generate an efficient image compression model. These approaches [21], [22] do not consider the variable computational complexity levels or bitrates. Therefore, they need to train and store one deep model for each bitrate and each computational complexity level, which is time-consuming and also increases the storage burden. In contrast, our CBANet can support variable bitrates under dynamic computational complexity constraints by using one single deep model.

### B. Variable Bitrate Image Compression

Several approaches [23] were proposed to achieve variable bitrate compression. For example, Choi et al. [24] proposed a conditional autoencoder structure to achieve variable bitrate image compression. Lu et al. [25] proposed the adaptive quantization layer for variable bitrate coding. Yang et al. [26] proposed to use modulated autoencoders for variable bitrate coding, while Lin et al. [27] introduced modulated generalized octave convolution and modulated internal feature maps in auto-encoder to achieve variable bitrate image compression.

### C. Adaptive Bitrate-Complexity-Coupled Image Compression

Recently, a few approaches [4], [5], [6] were also proposed to achieve the so-called adaptive bitrate-complexity-coupled image compression, in which the computational complexity level increases as the bitrate increases. For example, Toderici et al. [4], [5] proposed the recurrent neural network (RNN) based image compression systems, where the reconstructed images at different bitrates can be generated at different iterations in RNN (i.e., with different computational complexity levels). In a concurrent work [6], Yang et al. uses the SlimCAE structure to achieve the adaptive bitrate-complexity-coupled image compression. However, at each bitrate, these methods can only decode the bitstream under one fixed computational complexity level instead of supporting multiple computational complexity levels as achieved in our CBANet. For example, if we have three choices of computational complexity levels and three choices of bitrates, the methods [4], [5], [6] can only support three bitrate-complexity pairs (i.e., one complexity level is combined with one fixed bitrate). In contrast, our CBANet can support nine scenarios (i.e., we can support three complexity levels under each bitrate). Therefore, the goal in the methods [4], [5], [6] is substantially different from our approach. As a result, these methods do not decouple the rate-distortion-complexity optimization problem into two rate-distortion and complexity-distortion sub-problems as in our work, and the network design strategies in these methods are also different with our work, in which two modules are not introduced to respectively achieve rate-distortion and complexity-distortion trade-offs. To better understand our approach, we summarize the difference between our CBANet and the existing image compression methods in Table I. From Table I, our CBANet can support both adaptive bitrates and adaptive computational complexity levels, respectively.
complexity levels, which cannot be achieved by the existing learning-based deep image compression systems.

D. Dynamic Neural Networks

Many works were proposed to dynamically construct the networks in order to deploy CNNs at different scenarios. For example, Yu et al. [29] proposed a slimmable network structure to process the images by adaptively choosing the networks with different numbers of channels. Cai et al. [30] proposed the once-for-all network structure to deploy the network at different scenarios. Liu et al. [31] uses adaptive sampling for efficient 3D action recognition. However, these approaches only focus on the classification task and do not investigate the more challenging image compression task. In the image compression task, the input bitstreams will dynamically change if we aim to decode the bitstreams at various bitrates. If we simply apply these methods for the image compression task, we will require different CNNs at different bitrates. In contrast to these approaches, our CBANet can use a single CNN for decoding the bitstreams at multiple bitrates under various computational complexity constraints.

III. METHODOLOGY

In this section, we use the method [1] as an example to introduce our CBANet, which is the most fundamental structure for the learning-based image compression framework. We firstly introduce the overview of our CBANet. Then we will describe our bitrate adaptive module (BAM) and complexity adaptive module (CAM) in detail. After that, we will describe the training process of our CBANet. Finally, we will discuss the detailed structures of our CBANet based on other methods.

A. Overview

1) Formulation: For the learning-based image compression framework, the autoencoder structure is the most popular one [1], [2], [3], which consists of an encoder and a decoder. Given the input image \( x \), the encoder extracts the latent representation, which will then be quantized. The decoder takes the quantized representation as the input and uses a series of deconvolution operations to generate the reconstructed image. The goal of the existing learning-based image compression codec is to minimize the distortion \( D \) for any given bitrate \( R_{giv} \), i.e.,

\[
\min_{D} \quad s.t. \quad R \leq R_{giv},
\]

where \( R \) is the coding bitrate. This rate-distortion trade-off is usually solved by using the Lagrangian multiplier optimization technology, in which we usually train multiple models by using different pre-defined Lagrangian multiplier values.

However, the rate-distortion trade-off in Eq. (1) does not consider the computational complexity constraint. In contrast, we aim to minimize the following objective function in this work:

\[
\min_{D} \quad s.t. \quad R \leq R_{giv}, \quad C \leq C_{giv},
\]

where \( C \) represents the computational complexity of the current codec and \( C_{giv} \) is the given computational complexity constraint. It should be noted that both \( R_{giv} \) and \( C_{giv} \) are dynamically changed in this work.

2) Our Overall Pipeline: In this work, we focus on the decoder side, in which we aim to support multiple bitrates and computational complexity levels by using a single model. As shown in Fig. 2, our proposed CBANet consists of two modules: a bitrate adaptive module (BAM) and a complexity adaptive module (CAM). Specifically, the BAM consists of two parts: the bitrate adaptive layer (BAL) and the inverse bitrate adaptive layer (IBAL). The BAL aims to transfer the representation \( y^b \) at a base bitrate to that at the \( j \)-th supported bitrate \( y^j \). After the quantization operation, the quantized representation \( \hat{y}^j \) will be transmitted to the decoder side and we use the IBAL to transfer \( \hat{y}^j \) back to the representation at the base bitrate \( y^b \). Then we feed \( y^b \) to the CAM consisting of multiple branches, which can decode the images at different computational complexity levels by using different numbers of branches. Finally, our CBANet can support multiple bitrate decoding under various computational complexity levels through a single model. We will introduce more details about the BAM and the CAM below.

B. Bit Adaptive Module (BAM)

Our bitrate adaptive module (BAM) is used to achieve variable bitrate coding, which consists of two parts: the bitrate adaptive layer (BAL) and the inverse bitrate adaptive layer (IBAL). The BAL aims to transfer the representation from the base bitrate to that at the \( j \)-th supported bitrate for transmission, while the IBAL aims to transfer the transmitted representation from the \( j \)-th supported bitrate back to the base bitrate for the decoding process of CAM. To make our CBANet a general approach, we follow [25] to use a four-layer CNN structure to implement our BAL and IBAL. As shown in Fig. 2, given the input representation of the BAL \( y^b \), the output of BAL \( y^j \) can be written as follows:

\[
y^j = \begin{cases} 
  y^b & \text{when } R = R_{b}, \\
  y^b \cdot (1 - \text{Sigmoid}(B(y^b; \theta_{j}^{BAL}))) & \text{when } R \neq R_{b},
\end{cases}
\]

(3)

where \( B(\cdot, \cdot) \) is the function of our four-layer CNN in BAL and its parameters is denoted as \( \theta_{j}^{BAL} \) for the \( j \)-th supported bitrate. \( R \) is the bitrate of the input representation and \( R_b \) is the base bitrate.

Similar to our BAL, given the input representation of our IBAL \( \hat{y}^j \), the output representation of our IBAL \( \hat{y}^b \) can be written as follows:

\[
\hat{y}^b = \begin{cases} 
  \hat{y}^j & \text{when } R = R_{b}, \\
  \hat{y}^j \cdot (1 + \text{Relu}(B(\hat{y}^j; \theta_{j}^{IBAL}))) & \text{when } R \neq R_{b},
\end{cases}
\]

(4)

where \( \theta_{j}^{IBAL} \) denotes the parameters of the four-layer CNN in our IBAL for the \( j \)-th supported bitrate. The other notations are the same as those in Eq. (3). The four-layer CNN in our BAL and IBAL is illustrated in Fig. 3. The experiments show that we can achieve promising image compression performance by using a four-layer CNN and one Leaky Relu
Many existing approaches [1], [2]. Moreover, we also use the activation function in our CBANet. Therefore, we adopt this simple but effective design in our CBANet. Our CBANet is a general framework, and other bitrate adaptive methods like [32] for bitrate adaptation can also be readily incorporated into our framework.

C. Complexity Adaptive Module (CAM)

Fig. 4 shows the structure of our CAM, which consists of several parallel branches. The CNN in each branch directly takes the representation \( y^b \) as the input and generates one residual component of the reconstructed images. In each branch, we follow the method [1] to use the kernel size of \( 5 \times 5 \) and \( 9 \times 9 \), which is commonly adopted by many existing approaches [1], [2]. Moreover, we also use the inverse generalized divisive normalization (IGDN) layer [1] to normalize the output of each deconvolutional layer, which is commonly used in the existing approaches [1], [2]. Formally, the output of the \( i \)-th branch \( \tilde{x}_i \) can be formulated as follows:

\[
\tilde{x}_i = \mathcal{F}_i(y^b; \theta_i^{CAM}),
\]

where \( \mathcal{F}_i(\cdot, \cdot) \) is the function from the decoding process for the \( i \)-th branch in CAM and \( \theta_i^{CAM} \) denotes its parameters. After producing the output representation from all the branches, we aggregate the outputs from different branches by using the weighted sum of them, namely,

\[
\hat{x} = \sum_{i=1}^{K} g_i \cdot \tilde{x}_i,
\]

where the scalar \( g_i \) is the weight for the \( i \)-th branch. Since all the operations in Eq. (6) are differentiable, the scalar \( g_i \) can also be jointly learned with the parameters \( \theta_i^{CAM} \) in the \( i \)-th branch of our CAM in the training process. \( \tilde{x}_i \) is the scaled output of the \( i \)-th branch and \( K \) is the number of branches used in the decoding process. We have \( K \leq K_{max} \) where \( K_{max} = 3 \) in our implementation is the pre-defined total number of branches. In the decoding process, the computational complexity level will increase when the number of used branches \( K \) increases. \( \hat{x} \) is the final reconstructed image after aggregating the outputs from \( K \) branches. Based on the specific computational complexity constraint of the device/platform, we can automatically choose the number of used branches in the decoding process to generate the reconstructed images with different visual qualities.

D. Training Procedure

The training process of our CBANet consists of three stages: encoder preparation, CAM training, and BAM training. We will introduce them below.

1) Encoder Preparation: For preparation of the encoder, we follow the existing learning-based image compression methods like [1], [2], and [3] to train one pair of encoder-decoder at the base bitrate. Then, we use the learned encoder to generate the bitstream at the base bitrate \( y^b \). Note that once the encoder is trained at this stage, we will fix the encoder in the remaining training procedure. Our CBANet based decoder will replace the original decoder in order to support multiple bitrate decoding under various computational complexity constraints by using a single model.
After the encoder preparation stage is finished, we train our CBANet to support multiple computational complexity levels and bitrates. The trainable parameters consist of the parameters from both BAM and CAM. Note the trainable parameters in BAM include the parameters from both the BAL and the IBAL. It is a non-trivial task to solve the optimization problem in Eq. (2) by learning all the parameters in one step. Instead, in our proposed framework, we disentangle the parameters of these two modules and propose a two-step optimization procedure to respectively learn the parameters from CAM and BAM at each step.

2) CAM Training: At the first step, we propose to optimize the parameters in CAM by solving the following objective function related to the distortion and complexity trade-off when the bitrate is fixed:

$$\arg \min_{\theta_{CAM}^i} D, \quad \text{s.t.} \quad C = C_i, \quad R = R_b, \quad \forall \; i = 1, \ldots, N,$$

where $D$ denotes the distortion, $R$ and $C$ denote the bitrate and computational complexity level, respectively. $\theta_{CAM}^i$ is the parameters of the $i$-th branch in CAM, and $N$ is the total number of the supported computational complexity levels. $R_b$ and $C_i$ are the base bitrate and the $i$-th supported complexity level in our codec, respectively. In our implementation, we use the highest supported bitrate as our base bitrate.

In our implementation, we train the CAM by solving the objective function in Eq. (7) in a progressive manner. Specifically, we use the representation $\hat{y}_j$ at the base bitrate (i.e., $R = R_b$) as the input of the CAM and train the first branch by setting $K = 1$ in Eq. (6). In this case, we have $y_j = \hat{y}_j$ and $\hat{y}_b = \hat{y}_j$. We minimize the distortion between the reconstructed image and the input image and then back-propagate the gradients to update the parameters of the first branch $\theta_{CAM}^1$. After that, we fix the parameters of the first branch and train the second branch (i.e., we set $K = 2$ in Eq. (6)) in the same way as that for the first branch to learn the parameters in the second branch $\theta_{CAM}^2$. We repeat this procedure in a branch-by-branch fashion and learn the parameters in all branches of our complexity adaptive module. Finally, we obtain the learned CAM, which consists of the learned parameters from all the branches. In this process, we do not use the Lagrangian multiplier because we aim to learn the CAM to support multiple computational complexity levels at the base bitrate $R_b$.

3) BAM Training: At the second step, we fix the CAM and train the BAM by solving the following objective function related to the distortion and bitrate trade-off:

$$\arg \min_{\theta_{BAM}^j} D, \quad \text{s.t.} \quad C = C_b, \quad R \leq R_j, \quad \forall \; j = 1, \ldots, M - 1,$$

where $C_b$ is the base computational complexity level of our CAM and $R_j$ is the $j$-th supported bitrate. As our CBANet supports multiple bitrates and the quantized representations are different at different bitrates, we train one BAM for each bitrate except for the base bitrate (i.e., $R_b$ in Eq. (7)), and use $\theta_{BAM}^j$ to denote the parameters of the BAM for the $j$-th supported bitrate. $M$ is the total number of supported bitrates. We solve the optimization problem in Eq. (8) by using the Lagrangian multiplier optimization technology. Therefore, the optimization problem in Eq. (8) can be rewritten as:

$$\arg \min_{\theta_{BAM}^j} D + \lambda_j R, \quad \text{or equivalently,} \quad \arg \min_{\theta_{BAM}^j} R + \lambda_j D,$$

$$\text{s.t.} \quad C = C_b, \quad \forall \; j = 1, \ldots, M - 1,$$

where $\lambda_j = 1/\lambda_j$ and $\lambda_j$ is the $j$-th Lagrangian multiplier to produce the BAM that can support the $j$-th bitrate. The other notations are the same as those in Eq. (8). In our implementation, we use the highest supported computational complexity level as the base computational complexity level $C_b$. Note when $j = M$, we directly set $y_j = \hat{y}_b$ and $\hat{y}_b = \hat{y}_j$ by using the identity mapping in both the BAL and the IBAL.

In summary, when training the BAM, we feed the output of our encoder $\hat{y}_b$ to the subsequent BAL and IBAL to generate the representation $\hat{y}_b$ at the decoder side. Then we use $\hat{y}_b$ as the input of the CAM with all branches and produce the final reconstructed image. We optimize the objective function in Eq. (9) and back-propagate the gradients to update the parameters in BAM while keeping the parameters from the CAM and the encoder fixed. In this process, we use different Lagrangian multipliers in Eq. (9) to achieve variable bitrate coding, such that our BAM can perform bitrate adaptation after the training process. In this way, we can optimize the parameters in BAM in an end-to-end manner.

After the training process is finished, our BAM can transfer the representation $\hat{y}_b$ at the base bitrate to that at the $j$-th supported bitrate for transmission and transfer the transmitted representation $\hat{y}_b$ back to that at the base bitrate $\hat{y}_b$ for the decoding process of CAM. At the inference stage, we can support image compression at the target bitrate by using the corresponding learned BAM. Considering that the introduction of BAM only slightly increases the number of trainable parameters and the overall computational complexity (see Sec. IV for more details), our approach can support multiple bitrate setting without significantly increasing the storage requirement when deploying our CBANet in various deployment scenarios.

4) Discussion: The two objective functions in Eq. (7) and Eq. (8) are proposed based on the following two observations: (1) When the bitrate $R$ is fixed as the base bitrate (i.e., $R = R_b$), there is no reconstruction loss from the BAM as the output and the input of the BAM are the same (see Sec. III-B). Therefore, we focus on achieving the distortion and complexity trade-off in Eq. (7) without considering the influence from the BAM. (2) On the other hand, our CBANet at the highest computational complexity level achieves the best compression performance (see Sec. IV for more details). Therefore, we use the highest computational complexity level as our base computational complexity level $C_b$ and then train the BAM to focus on achieving the rate-distortion trade-off in Eq. (8), in which the reconstruction loss from the CAM will not significantly affect the training process of the BAM. In summary, instead of directly solving Eq. (2) in this work,
we achieve an alternative rate-distortion-complexity trade-off by using a two-step optimization strategy. Our training strategy disentangles the two sets of parameters and thus the training process for learning one set of parameters (i.e., parameters of CAM or parameters of BAM) is relatively independent and will not be substantially affected by the reconstruction loss from another module.

E. Structures of Our CBANet Based on Other Methods

1) Structure of CBANet-HP: Our CBANet Based on [2]:
For better presentation, we name our CBANet based on the baseline method Ballé et al. [2] as CBANet-HP. Fig. 5 shows the network structure of our CBANet-HP. Similar to the encoder in our CBANet, the parameters of the hyperprior part (i.e., Hyper Encoder and Hyper Decoder in Fig. 5) in our CBANet-HP are directly copied from the learned baseline model at the base bitrate, namely, they are jointly learned with the encoder at the encoder preparation stage. Once the hyperprior part is trained, we fix this part in the subsequent training procedure. Since the hyperprior part is learned by training the baseline method [2] at the base bitrate, it may not achieve promising performance when its input representation is at a target bitrate (e.g., $y_j$ at the $j$-th supported bitrate). Therefore, we additionally introduce one IBAL before “Hyper Encoder” to convert the representation back to that at the base bitrate to compensate the representation mismatch. Similarly, the output of “Hyper Decoder” may not generate useful entropy model information for the arithmetic encoder and the arithmetic decoder when the representation is at the target bitrate (e.g., $\hat{y}_j$ at the $j$-th supported bitrate). Therefore, we additionally introduce one BAL after “Hyper Decoder” to generate reasonable entropy model information for the representation at the target bitrate (e.g., $\hat{y}_j$ at the $j$-th support bitrate).

Different from the structure introduced in Fig. 4, in our CBANet-HP, each branch of the CAM consists of four deconvolutional layers, and the kernel size of each deconvolutional layer is $5 \times 5$. The other settings are the same as those in Fig. 4.

2) Structure of CBANet-AR: Our CBANet Based on [3]:
For better presentation, we name our CBANet based on the baseline method Minnen et al. [3] as CBANet-AR. Fig. 6 shows the structure of our CBANet-AR. Similar to our CBANet-HP, the parameters of the hyperprior part (i.e., “Hyper Encoder” and “Hyper Decoder”) in Fig. 6) and the context part (i.e., “Context Model” and “Entropy Parameters”) in Fig. 6) in our CBANet-AR are directly copied from the learned baseline model at the base bitrate. We also fix these two parts for the subsequent training procedure. Since the method in [3] uses a complex structure to generate the entropy model information for the arithmetic encoder and the arithmetic decoder, which includes the context part and the hyperprior part, we additionally introduce two BAL (i.e., one BAL between “Context Model” and “Entropy Parameters”, and one BAL between “Hyper Decoder” and “Entropy Parameters”) in our CBANet-AR to facilitate its training process. The introduction of these two BAL only slightly increases the cost of our CBANet-AR (see Sec. IV for more details), but it can lead to better compression performance.

Similar to our CBANet-HP, each branch of the CAM in our CBANet-AR consists of four deconvolutional layers, and the kernel size of each deconvolutional layer is $5 \times 5$.

3) Structure of CBANet-GA: Our CBANet Based on [7]:
We also build our CBANet based on Cheng et al. [7] and name this method as CBANet-GA. Fig. 7 shows the network structure of our CBANet-GA. We follow the design in Cheng et al. [7] to use the same number of layers and the same kernel size at each layer in our CAM.

IV. Experiments

To demonstrate the effectiveness of our CBANet, we implement our framework based on three popular learning-based
image compression methods: Ballé et al. [2] (i.e., CBANet-HP), Minnen et al. [3] (i.e., CBANet-AR), and Cheng et al. [7] (i.e., CBANet-GA). We do not implement our CBANet based on the method [1] because its performance is not state-of-the-art. We perform the experiments on two benchmark datasets: Kodak [33] and Workshop and Challenge on Learned Image Compression (CLIC) [34]. In this section, we use the number of floating point operations (#FLOPs) as the criterion for computational complexity measurement, which is commonly used in recent works [35], [36], [37], [38], [39], [40]. Since we focus on the decoder side, we only report the #FLOPs used in the decoding process when evaluating the computational complexity.

1) Datasets: The Kodak dataset [33] consists of 24 uncompressed images with the resolution of 768 × 512. For the CLIC dataset [34], we use the professional dataset in CLIC2020 competition in our experiment, which consists of 41 images.

A. Experiments on Kodak

1) Implementation Details: Our CBANet is trained by using one Nvidia 2080Ti GPU. We use 20,745 high-quality images from Flickr.com and take the randomly cropped patches with the resolution of 256 × 256 as the training data.

For the baseline method Ballé et al. [2], we follow [2] to use 128 channels in the encoder for the four low bitrates and train the baseline model at the base bitrate by using the Lagrangian multiplier $\lambda$ of 2048. We empirically set the number of channels $P$ in BAM as 192. In the CAM, we set the total number of branches $K_{\text{max}}$ as 3. We respectively set the numbers of channels $T$ in Branch1, Branch2, and Branch3 of the CAM as 60, 60, and 88 for low bitrates, such that the #FLOPs in Branch1, Branch2, and Branch3 take 25%, 25%, and 50% of the total #FLOPs, respectively. For three high bitrates, we follow [2] to use 192 channels in the encoder and train the baseline model at the base bitrate by using the Lagrangian multiplier $\lambda$ of 8192. The number of channels $P$ in BAM is empirically set as 320. We respectively set the numbers of channels $T$ in Branch1, Branch2, and Branch3 of the CAM as 90, 90, and 132 for high bitrates, such that the #FLOPs in Branch1, Branch2, and Branch3 take 25%, 25%, and 50% of the total #FLOPs, respectively. We use the Adam optimizer for optimization. The learning rate and the batch size are respectively set as $1e^{-4}$ and 8 for all the experiments.

Similar to the experiments for CBANet-HP, we train the model at the base bitrate by using the Lagrangian multiplier $\lambda$ of 8192 for the baseline method Minnen et al. [3]. We follow [3] to use 320 channels in the encoder at the base bitrates. The number of channels $P$ in BAM is empirically set as 320. We respectively set the numbers of channels $T$ in Branch1, Branch2, and Branch3 as 136, 159, and 230, such that the #FLOPs in Branch1, Branch2, and Branch3 take 25%, 25%, and 50% of the total #FLOPs, respectively. The other settings are the same as those for CBANet-HP except that we use the learning rate of $1e^{-5}$ when training the BAM.

For CBANet-GA, we follow [7] to use 192 channels in the encoder for low bitrates and train the baseline method at the base bitrate by using the Lagrangian multiplier $\lambda$ of 512. We empirically set the number of channels $P$ as 320. We respectively set the number of channels $T$ in Branch1, Branch2, and Branch3 of the CAM as 43, 48, and 67, such that the #FLOPs in Branch1, Branch2, and Branch3 take 25%, 25%, and 50% #FLOPs, respectively. For high bitrates, we follow [7] to use 256 channels in the encoder and train the baseline method at the base bitrate by using the Lagrangian multiplier $\lambda$ of 2048. We respectively set the number of channels $T$ in Branch1, Branch2, and Branch3 of the CAM as 57, 63, and 128, such that the #FLOPs in Branch1, Branch2, and Branch3 takes 25%, 25%, and 50% #FLOPs, respectively. The other settings are the same as those for low bitrates.

2) Experimental Results: In Table II, we report the BDBR and BD-PSNR values [41] when comparing our CBANet-HP (resp., CBANet-AR, CBANet-GA) under multiple computational complexity levels with the baseline algorithm Ballé et al. [2] (resp., Minnen et al. [3], Cheng et al. [7]) on the Kodak dataset. Our CBANet-HP (resp., CBANet-AR, CBANet-GA) when using one, two and three branches in the decoding process are denoted as CBANet-HP-25% (resp., CBANet-AR-25%, CBANet-GA-25%), CBANet-HP-50% (resp., CBANet-AR-50%, CBANet-GA-50%) and CBANet-HP-100% (resp., CBANet-AR-100%, CBANet-GA-100%), respectively. From Table II, we have the following observations: (1) For all the three methods (i.e., CBANet-HP, CBANet-AR, and CBANet-GA), the BD-PSNR value increases as the computational complexity level increases (i.e., the number of branch increases), which demonstrates the effectiveness of the proposed complexity adaptive module. For example, the BD-PSNR values are $-0.18$, $0.15$ and $0.27$ when comparing our CBANet-GA-25%, CBANet-GA-50%, and CBANet-GA-100% with the baseline method Cheng et al. [7], respectively. (2) When comparing our CBANet-HP-100% with the baseline method Ballé et al. [2], although the BD-PSNR value decreases by $0.13$dB, our CBANet-HP can support multiple bitrates and various computational complexity levels by using a single network, which can help us to deploy the learning-based image compression systems in different deployment scenarios. (3) The BD-PSNR value even increases by
Fig. 8. PSNR comparison between the baseline method Ballé et al. (2018) [2] and our CBANet-HP at different computational complexity levels on the Kodak dataset.

Fig. 9. MS-SSIM comparison of our CBANet-HP and the baseline method Ballé et al. (2018) [2] on the Kodak dataset.

Fig. 10. PSNR comparison of different image compression methods on Kodak.

0.01dB for our CBANet-AR-100% when compared with the baseline method Minnen et al. [3], which further demonstrates the effectiveness of our CBANet for image compression. (4) The BD-PSNR values increase when comparing our CBANet-GA-50% and CBANet-GA-100% with the baseline method Cheng et al. [7], which shows our CBANet is a general framework that can effectively build based on different baseline methods.

3) RD Curves: In Fig. 8, we take our CBANet-HP as an example and report the RD curves when comparing our CBANet-HP with the baseline method Ballé et al. (2018) [2] in terms of peak signal-to-noise ratio (PSNR) on the Kodak dataset. We have similar observations for our CBANet-AR and CBANet-GA methods. From Fig. 8, we have the following observations: (1) When using the highest computational complexity level, our CBANet-HP-100% achieves similar performance when compared with the baseline algorithm [2] at most bitrates. It demonstrates the effectiveness of our CBANet-HP for image compression. (2) Although the average performance of our CBANet-HP-25% drops about 0.3dB when compared with the baseline method [2], we can significantly reduce the computational complexity by about 75% to accelerate the decoding process. (3) The performance of our CBANet-HP drops by about 0.5dB at bpp = 0.28. We hypothesize that this is mainly because bpp = 0.28 is relatively far from bpp = 0.77, which is the base bitrate in our implementation. Therefore, it is relatively hard to convert the representation at the base bitrate to that at bpp = 0.28. In other words, the BAM is the bottleneck that limits the performance of our CBANet-HP when setting bpp = 0.28, which is verified by the results that our CBANet-HP-25%, CBANet-HP-50%, and CBANet-HP-100% achieve similar performance when setting bpp = 0.28.

We compare the MS-SSIM results of our CBANet-HP with the baseline method Ballé et al. (2018) [2]. The results are shown in Fig. 9, from which we have similar observations as those for PSNR. Therefore, we do not provide further analysis here.

We also compare our CBANet with the state-of-the-art methods. Specifically, in Fig. 10, we compare the performance of our CBANet-GA-100% with the baseline methods Hu et al. (2020) [20], Akbari et al. (2021) [28], BPG [10], HEVC-Intra [42], and VVC-Intra [43] on the Kodak dataset. For BPG, we report the results using the mode YUV444 (BPG444). For HEVC-Intra, we report the results when using the model HM 16.25 with the mode YUV444. For VVC-Intra, we report the results using the model VTM 12.1 with the mode YUV444. From Fig. 10, our CBANet-GA-100% achieves comparable performance with the method VTM 12.1 and outperforms other baseline methods.

4) Comparison of Storage Requirement: In Table III, we compare the storage requirement of our CBANet-HP (resp., CBANet-AR, CBANet-GA) and the baseline method Ballé et al. [2] (resp., Minnen et al. [3], Cheng et al. [7]) when supporting multiple bitrates and three computational complexity levels.

Taking the comparison between our CBANet-HP and the baseline method Ballé et al. [2] as an example, the number of parameters of the baseline method [2] for supporting one single bitrate and one fixed computational complexity level is 2.53M. One straightforward way to support multiple computational complexity levels is to reduce the number of channels for the baseline method to satisfy different computational complexity constraints and store one deep model for each computational complexity level. When calculating the
The number of parameters of our CBANet-HP is 3.35M.

We need to store one single model with three BAM to support various computational complexity levels setting. In contrast, we only need to store one single model with three BAM (the number of parameters is 5.93M) to support four bitrates, the corresponding number of parameters increases significantly. Specifically, for each single bitrate and three computational complexity levels, the total number of parameters is 5.93M at each single bitrate, which includes 1.52M, 1.88M and 2.53M for storing the models with 60, 88 and 128 channels, respectively.

Furthermore, when variable bitrate coding is required in practical applications, the baseline method [2] has to train different models for different bitrates. Therefore, the total number of parameters increases significantly. Specifically, for the single bitrate and three computational complexity levels setting, the number of parameters is 5.93M. If we want to support four bitrates, the corresponding number of parameters becomes 23.72M (=5.93 × 4) for four bitrates and three computational complexity levels setting. In contrast, we only need to store one single model with three BAM (the number of parameters of each BAM is only 0.17M) to support various bitrates under different computational complexity constraints.

The number of parameters of our CBANet-HP is 3.35M (=2.84M + 0.17M × (4 − 1)), which achieves a 85.9% reduction in terms of the number of parameters.

5) Practical Speedup: In order to demonstrate the practical speedup after using our proposed method, we report the practical running time of our CBANet-HP and the baseline method Ballé et al. [2], Yang et al. [26] and Lin et al. [27]. For the method Yang et al. [26], we report the decoding time with the hyperprior as it achieves better compression performance. Using the machine with one Intel Core i5-7500 CPU, the average latency of these baseline methods Ballé et al. [2], Yang et al. [26], Lin et al. [27], CBANet-HP-25%, CBANet-HP-50% and CBANet-HP-100% are 633.26ms, 662.03ms, 5748.40ms, 204.59ms, 391.64ms and 629.94ms for decoding one single image on Kodak, respectively. From the results, our CBANet-HP-25% achieves 67.69% speedup when compared with the baseline method [2], and requires less decoding time when compared with the baseline methods [26] and [27].

In Table IV, we provide the BDBR and BD-PSNR values [41] when comparing our CBANet-HP (resp., CBANet-AR, CBANet-GA) with the baseline algorithm Ballé et al. [2] (resp., Minnen et al. [3], Cheng et al. [7]) on the CLIC dataset. We have similar observations as those on the Kodak dataset. Therefore, we do not provide further discussion here.

B. Experiments on CLIC

In Table IV, we provide the BDBR and BD-PSNR values [41] when comparing our CBANet-HP (resp., CBANet-AR, CBANet-GA) with the baseline algorithm Ballé et al. [2] (resp., Minnen et al. [3], Cheng et al. [7]) on the CLIC dataset. We have similar observations as those on the Kodak dataset. Therefore, we do not provide further discussion here.

C. Ablation Study and Algorithm Analysis

1) Effectiveness of Our BAM: To demonstrate the effectiveness of our BAM for transferring the representation from the base bitrate to those at different bitrates, we take the method Ballé et al. (2018) [2] as an example and perform the experiments on the Kodak dataset to only transfer the representation without introducing the CAM in our CBANet-HP to support multiple bitrates. In this case, we firstly train an encoder-decoder pair at the base bitrate by using the method [2]. Then we introduce our BAM in this codec for variable bitrate image compression without replacing the decoder with the CAM. The result is referred to as Ballé et al. (2018)-VB in Fig. 11. We observe that the Ballé et al. (2018)-VB method is comparable with the baseline method [2] at most bitrates, which demonstrates that it is effective to use our BAM to support variable bitrate image compression. Moreover, the performance of the Ballé et al. (2018)-VB method drops about 0.4dB at bpp = 0.30 when compared with the baseline.

### Table III

| #Supported bitrates | 1       | n       |
|---------------------|---------|---------|
| Ballé et al. (2018) | 2.53M   | (2.53n)M|
| (for one complexity level) |         |         |
| Ballé et al. (2018) | 5.93M   | (5.93n)M|
| (for three complexity levels) |         |         |
| CBANet-HP (ours)   | 2.84M   | [2.84+0.17(n-1)]M|
| (for three complexity levels) |         |         |
| Minnen et al. (2018) | 25.01M | (25.01n)M|
| (for one complexity level) |         |         |
| Minnen et al. (2018) | 65.28M | (65.28n)M|
| (for three complexity levels) |         |         |
| CBANet-AR (ours)  | 26.35M  | [26.35+1.73(n-1)]M|
| (for three complexity levels) |         |         |
| Cheng et al. (2020) | 13.48M  | (13.48n)M|
| (for one complexity level) |         |         |
| Cheng et al. (2020) | 31.77M  | (31.77n)M|
| (for three complexity levels) |         |         |
| CBANet-GA (ours) | 13.31M  | [13.31+1.80(n-1)]M|
| (for three complexity levels) |         |         |

### Table IV

| Method | BDBR(%) | BD-PSNR(dB) |
|--------|---------|-------------|
| CBANet-HP-25% | 4.71 | -0.22 |
| CBANet-HP-50% | 3.60 | -0.17 |
| CBANet-HP-100% | 1.20 | -0.05 |
| CBANet-AR-25% | -1.07 | 0.04 |
| CBANet-AR-50% | -2.39 | 0.10 |
| CBANet-AR-100% | -2.97 | 0.13 |
| CBANet-GA-25% | 0.69 | -0.06 |
| CBANet-GA-50% | -5.75 | 0.21 |
| CBANet-GA-100% | -8.42 | 0.32 |
method [2], which again demonstrates that the BAM limits the performance of our CBANet-HP at the lowest bitrate.

We also report the #FLOPs of the BAM in our CBANet-HP. Taking our CBANet-HP-100% at the low bitrates as an example, the #FLOPs of the BAM and our CBANet-HP-100% are 0.26G and 61.52G, respectively. The #FLOPs of our BAM only takes 0.42% of the total #FLOPs, which indicates that our CBANet-HP can support more bitrates by slightly increasing the computation cost.

2) Comparison With the Alternative Method by Training From Scratch: One may ask the question: How is the performance if we directly train the codec with a low complexity decoder (i.e., with less #FLOPs) by using the existing methods? To address this concern, we take the method Ballé et al. (2018) [2] as an example and perform the experiment to use less #FLOPs in the decoder by setting the number of channels in the decoder as 58. We set the encoder structure the same as that in the baseline method [2]. In this case, the #FLOPs for the decoder is on par with our CBANet-HP-25%. We directly train the codec with the low complexity decoder from scratch and the result is referred to as Ballé et al. (2018)-25% in Fig. 11. From the results, we observe that the performance of the Ballé et al. (2018)-25% method is comparable with our CBANet-HP-25% approach. However, we would like to highlight that it takes 3,000,000 iterations for the Ballé et al. (2018)-25% method to learn one model at each bitrate. On the other hand, it only takes 300,000 iterations in our CBANet-HP-25% to learn one BAM for supporting one additional bitrate after we train the model at the base bitrate, which is $10 \times$ less than that in the Ballé et al. (2018)-25% method.

3) Effect of the Total Number of Branches $K_{\text{max}}$: To investigate the effect of the total number of branches $K_{\text{max}}$ in our CBANet, we take our CBANet-HP method on the Kodak dataset as an example and perform the experiments to use less #FLOPs in the decoder by setting the number of channels in these four branches such that each branch takes 25% of the total #FLOPs. Our method when using one, two, three, and four branches are named as CBANet-HP-25% ($K_{\text{max}} = 4$), CBANet-HP-50% ($K_{\text{max}} = 4$), CBANet-HP-75% ($K_{\text{max}} = 4$), and CBANet-HP-100% ($K_{\text{max}} = 4$) in Fig. 12(b), respectively. In Fig. 12(c), we report the performance of our CBANet-HP when setting $K_{\text{max}} = 5$. In this case, we have five branches in our CBANet-HP, which can support four computational complexity levels. We adjust the numbers of channels in these five branches such that each branch takes 20% of the total #FLOPs. Our method when using one, two, three, four, and five branches are named as CBANet-HP-20% ($K_{\text{max}} = 5$), CBANet-HP-40% ($K_{\text{max}} = 5$), CBANet-HP-60% ($K_{\text{max}} = 5$), CBANet-HP-80% ($K_{\text{max}} = 5$), and CBANet-HP-100% ($K_{\text{max}} = 5$) in Fig. 12(c), respectively.

From Fig. 12, when setting $K_{\text{max}}$ as 2, 4, and 5, we have similar observations as those when setting $K_{\text{max}}$ as 3, which indicates that the performance of our CBANet is not sensitive to the total number of branches $K_{\text{max}}$. Therefore, our CBANet can support more computational complexity levels by increasing the total number of branches $K_{\text{max}}$. In our experiment, we set $K_{\text{max}} = 3$ by default and use it as an example to demonstrate the effectiveness of our CBANet.

4) Effect of Computational Complexity Allocation at Different Branches: To investigate the effect of allocating different computational complexity at different branches, we take our CBANet-HP under the computational complexity level of 50% FLOPs on the Kodak dataset as an example and perform additional experiments to allocate different computational complexity at different branches. For the CBANet-HP-50% (50%) method, we train one branch with 50% FLOPs in our CBANet-HP, while for the CBANet-HP-50% (20% + 30%) approach, we use two branches in our CBANet-HP to achieve 50% FLOPs, in which the first and the second branches take 20% and 30% FLOPs, respectively. In the CBANet-HP-50% method, the first and the second branches take 25% and 25% FLOPs, respectively, which is our default setting. In Table V, we report the BDBR and BD-PSNR values when comparing the methods CBANet-HP-50% (50%), CBANet-HP-50% (20% + 30%), and CBANet-HP-50% with the baseline method Ballé et al. [2] on the Kodak dataset.

![Graph](image-url)
the methods CBANet-HP-50% (50%), CBANet-HP-50% (20% + 30%), and our CBANet-HP-50% with the baseline method Ballé et al. (2018) [2]. From Table V, the results of the three methods CBANet-HP-50% (50%), CBANet-HP-50% (20% + 30%), and CBANet-HP-50% are comparable, which demonstrates that the performance of our CBANet is not sensitive to the strategy on how to allocate computational complexity at different branches under the given overall computational complexity.

5) Visualization of the Reconstructed Images Generated by Using Our CBANet Under Different Computational Complexity Levels: In Fig. 13, we take our CBANet-HP as an example and provide the visualization results of two images from internet. The reconstructed images in Fig. 13 are generated by using our CBANet-HP under different computational complexity levels. Moreover, we also visualize the residuals between these reconstructed images to better illustrate their differences. From Fig. 13, the reconstructed images generated by using our CBANet-HP-100% have fewer artifacts at the edge of the objects when compared with those generated by using our CBANet-HP under lower computational complexity levels (e.g., CBANet-HP-25%), which further demonstrates the effectiveness of our CBANet for image compression.

6) Visualization of the Reconstructed Images Generated by Using Different Image Compression Methods: In Fig. 14, we take our CBANet-HP as an example and provide the visualization results of two images from internet. The reconstructed images in Fig. 14 are generated by using the JPEG, JPEG...
In this work, we have introduced a new deep image compression framework that aims to learn one single network to achieve complexity and bitrate adaptive deep image compression, for which we propose a new objective function that considers the rate-distortion-complexity trade-off. We also propose a two-step approach to decouple the complex rate-distortion-complexity optimization problem into a complexity-distortion optimization sub-task and a rate-distortion optimization sub-task. In addition, a new network design strategy is also proposed, in which we introduce a complexity adaptive module (CAM) and a bitrate adaptive module (BAM) to respectively achieve complexity-distortion and rate-distortion trade-offs. Different from the existing learning-based deep image compression approaches that need to train different models for different bitrate levels, our Complexity and Bitrate Adaptive Network (CBANet) can support variable bitrate under various computational complexity levels, our Complexity and Bitrate Adaptive Network (CBANet) can support variable bitrate compression approaches, which demonstrates that it is beneficial to use our CBANet to compress the images.

V. CONCLUSION

In this work, we have introduced a new deep image compression framework that aims to learn one single network to achieve complexity and bitrate adaptive deep image compression, which demonstrates that it is beneficial to use our CBANet to compress the images.

2000, and our CBANet-HP-100% methods. From Fig. 14, our CBANet-HP-100% method performs better than the JPEG and JPEG 2000 approaches, which demonstrates that it is beneficial to use our CBANet to compress the images.

REFERENCES

[1] J. Ballé, V. Laparra, and E. Simoncelli, “End-to-end optimized image compression,” in Proc. ICLR, 2017, pp. 1–27.
[2] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior,” in Proc. ICLR, 2018, pp. 1–23.
[3] D. Minnen, J. Ballé, and G. D. Toderici, “Joint autoregressive and hierarchical priors for learned image compression,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 10771–10780.
[4] G. Toderici et al., “Variable rate image compression with recurrent neural networks,” in Proc. ICLR, 2016, pp. 1–12.
[5] G. Toderici et al., “Full resolution image compression with recurrent neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017.
[6] F. Yang, L. Herranz, Y. Cheng, and M. G. Mozerv, “Slimmable compressive autoencoders for practical neural image compression,” 2021, arXiv:2103.15726.
[7] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, “Learned image compression with discretized Gaussian mixture likelihoods and attention modules,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 7939–7948.
[8] G. K. Wallace, “The JPEG still picture compression standard,” Commun. ACM, vol. 34, no. 4, pp. 30–44, Apr. 1991.
[9] A. Skodras, C. Christopoulos, and T. Ebrahimi, “The JPEG 2000 still image compression standard,” IEEE Signal Process. Mag., vol. 18, no. 5, pp. 36–58, Sep. 2001.
[10] F. Bellard. (2018). BPG Image Format. [Online]. Available: https://bellard.org/bpg/
[11] C. E. Shannon, “A mathematical theory of communication,” ACM SIGMOBILE Mobile Comput. Commun. Rev., vol. 5, no. 1, pp. 3–55, Jan. 2001.
[12] D. Minnen, G. Toderici, S. Singh, S. J. Hwang, and M. Covell, “Image-dependent local entropy models for learned image compression,” in Proc. 25th IEEE Int. Conf. Image Process. (ICIP), Oct. 2018, pp. 430–434.
[13] E. Agustsson et al., “Soft-to-hard vector quantization for end-to-end learning compressible representations,” in Proc. NeurIPS, 2017, pp. 1–11.
[14] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” in Proc. ICLR, 2017, pp. 1–19.
[15] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, “Conditional probability models for deep image compression,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4394–4402.
[16] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, “Generative adversarial networks for extreme learned image compression,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 221–231.
[17] J. Lee, S. Cho, and S.-K. Beack, “Context-adaptive entropy model for end-to-end optimized image compression,” in Proc. ICLR, 2019, pp. 1–20.
[18] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, “Learning convolutional networks for content-weighted image compression,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018.
[19] D. Minnen and S. Singh, “Channel-wise autoregressive entropy models for learned image compression,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Oct. 2020.
[20] Y. Hu, W. Yang, and J. Liu, “Coarse-to-fine hyper-prior modeling for learned image compression,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 11013–11020.
[21] N. Johnston, E. Eban, A. Gordon, and J. Ballé, “Computationally efficient neural image compression,” 2019, arXiv:1912.08771.
[22] O. Rippel and L. Bourdev, “Real-time adaptive image compression,” in Proc. ICML, 2017, pp. 2922–2930.
[23] N. Johnston et al., “Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018.

[24] Y. Choi, M. El-Khamy, and J. Lee, “Variable rate deep image compression with a conditional autoencoder,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 3146–3154.

[25] G. Lu, X. Zhang, W. Ouyang, L. Chen, Z. Gao, and D. Xu, “An end-to-end learning framework for video compression,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 10, pp. 3292–3308, Oct. 2021.

[26] F. Yang, L. Herranz, J. Van De Weijer, J. A. I. Guitián, A. M. López, and M. G. Mozeroz, “Variable rate deep image compression with modulated autoencoder,” IEEE Signal Process. Lett., vol. 27, pp. 331–335, 2020.

[27] J. Lin et al., “Learned variable-rate multi-frequency image compression using modulated generalized octave convolution,” 2020, arXiv:2009.13074.

[28] M. Akbari, J. Liang, J. Han, and C. Tu, “Learned multi-resolution variable-rate image compression with octave-based residual blocks,” IEEE Trans. Multimedia, vol. 23, pp. 3013–3021, 2021.

[29] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, “Slimmable neural networks,” in Proc. ICLR, 2019, pp. 1–12.

[30] H. Cai, C. Gan, and S. Han, “Once for all: Train one network and specialize it for efficient deployment,” in Proc. ICLR, 2020, pp. 1–15.

[31] J. Liu, J. Guo, and D. Xu, “APSNet: Toward adaptive point sampling for efficient 3D action recognition,” IEEE Trans. Image Process., vol. 31, pp. 5287–5302, 2022.

[32] Z. Cui, J. Wang, S. Gao, T. Guo, Y. Feng, and B. Bai, “Asymmetric gained deep image compression with continuous rate adaptation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021.

[33] E. Kodak. (1993). Kodak Lossless True Color Image Suite (PHOTOCDO992). [Online]. Available: http://r0k.us/graphics/kodak

[34] G. Toderici et al. (2020). Workshop and Challenge on Learned Image Compression. [Online]. Available: http://challenge.compression.cc/tasks/

[35] J. Guo, W. Zhang, W. Ouyang, and D. Xu, “Model compression using progressive channel pruning,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 3, pp. 1114–1124, Mar. 2020.

[36] J. Guo, W. Ouyang, and D. Xu, “Multi-dimensional pruning: A unified framework for model compression,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1508–1517.

[37] J. Liu, J. Guo, and D. Xu, “GeometryMotion-Transformer: An end-to-end framework for 3D action recognition,” IEEE Trans. Multimedia, pp. 1–13, 2022, doi: 10.1109/TMM.2022.3198011.

[38] J. Guo, W. Ouyang, and D. Xu, “Channel pruning guided by classification loss and feature importance,” in Proc. AAAI, 2020, pp. 10885–10892.

[39] J. Guo, J. Liu, and D. Xu, “JointPruning: Pruning networks along multiple dimensions for efficient point cloud processing,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 6, pp. 3659–3672, Jun. 2021.

[40] J. Guo, J. Liu, and D. Xu, “3D-Pruning: A model compression framework for efficient 3D action recognition,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 12, pp. 8717–8729, Dec. 2022.

[41] G. Bjontegaard, Calculation of Average PSNR Differences Between RD-Curves, document VCEG-M33, 2001.

[42] JVE Team. HM Reference Software for HEVC. Accessed: Sep. 21, 2022. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jve/HM

[43] JVE Team. VVC Official Test Model VTM. Accessed: Sep. 22, 2022. [Online]. Available: https://vcgit.hhi.fraunhofer.de/v entreg/vtmt12.

[44] Python Imaging Library (PIL). Accessed: May 16, 2021. [Online]. Available: https://pillow.readthedocs.io/en/stable/

[45] Jpeg2000 Official Software Openjepg. Accessed: May 16, 2021. [Online]. Available: https://github.com/uclouvain/openjepg

Jinyang Guo received the B.E. degree (Hons.) from the University of New South Wales in 2017 and the Ph.D. degree from The University of Sydney in 2022. He is currently an Assistant Professor with the Institute of Artificial Intelligence, Beihang University. His works have been published on top-tier journals and conferences (e.g., IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), CVPR, and AAAI). His research interests include efficient machine/deep learning methods. He serves as a Reviewer/Program Committee Member for IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), International Journal of Computer Vision (IJCV), CVPR, and ICCV.

Dong Xu (Fellow, IEEE) received the B.E. and Ph.D. degrees from the University of Science and Technology of China, in 2001 and 2005, respectively. While pursuing the Ph.D. degree, he was an Intern at Microsoft Research Asia, Beijing, China, and a Research Assistant at the Chinese University of Hong Kong, Shatin, Hong Kong, for more than two years. He was a Postdoctoral Research Scientist at Columbia University, New York, NY, USA, for one year. He also worked as a Faculty Member at Nanyang Technological University, Singapore, and the Chair of computer engineering at The University of Sydney, Australia. He is currently a Professor with the Department of Computer Science, The University of Hong Kong, Hong Kong. His current research interests include computer vision, statistical learning, and multimedia content analysis. He was a coauthor of a paper that received the Best Student Paper Award in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) in 2010, and a paper that received the Prize Paper Award in IEEE TRANSACTIONS ON MULTIMEDIA (T-MM) in 2014.

Guo Lu (Member, IEEE) received the B.S. degree from Ocean University of China in 2014 and the Ph.D. degree from Shanghai Jiao Tong University in 2020. Currently, he is an Assistant Professor with the School of Electronics Information and Electrical Engineering, Shanghai Jiao Tong University, China. His research interests include image and video processing, video compression, and computer vision. His works have been published in top-tier journals and conferences (e.g., IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), CVPR, and ECCV). He serves as a Reviewer/Program Committee Member for IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), T-TP, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (T-CSVT), and AAAI. He serves as the Guest Editor for International Journal of Computer Vision (IJCV) and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (T-CSVT).