# A unified efficient deep image compression framework and its application on human-centric Task

Xueyuan Chen[1] · Zhihao Hu[1] · Guo Lu[2] · Jiaheng Liu[1]

## Abstract

Image compression is a widely used technique to reduce the spatial redundancy in images. Recently, learning based image compression has achieved significant progress by using the powerful representation ability from neural networks. However, the current learning based image compression methods suffer from the huge computational cost, which limits their capacity for practical applications. In this paper, we propose a unified framework called Efficient Deep Image Compression (EDIC) based on three new technologies, including a channel attention module, a Gaussian mixture model and a decoder-side enhancement module. Specifically, we design an auto-encoder style network for learning based image compression. To improve the coding efficiency, we exploit the channel relationship between latent representations by using the channel attention module. Besides, the Gaussian mixture model is introduced for the entropy model and improves the accuracy for bitrate estimation. Furthermore, we introduce the decoder-side enhancement module to further improve image compression performance. Our EDIC method can also be readily incorporated with the Deep Video Compression (DVC) framework (Lu et al. 2019) to further improve the video compression performance. Simultaneously, our EDIC method boosts the coding performance significantly while bringing slightly increased computational cost. More importantly, experimental results demonstrate that the proposed approach outperforms the current image compression methods and is up to more than 150 times faster in terms of decoding speed when compared with Minnen's method (Minnen et al. 2018). Moreover, we also evaluate the performance of the human-centric task (i.e., face recognition) by using different coding strategies.

**Keywords** Image compression · Neural network · Auto-encoder · Gaussian mixture model

✉ Jiaheng Liu
liujiaheng@buaa.edu.cn

Xueyuan Chen
xueyuanchen@buaa.edu.cn

Zhihao Hu
huzhihao@buaa.edu.cn

Guo Lu
luguo2014@sjtu.edu.cn

1    Beihang University, Beijing, China

2    Shanghai Jiao Tong University, Shanghai, China

## 1 Introduction

Image compression aims to reduce the spatial redundancy in images and is widely used to save bandwidth and storage sizes in lots of applications. Traditional image compression methods [2, 3, 41, 46] rely on hand-crafted techniques to improve compression efficiency. For example, JPEG [46] uses the discrete cosine transform (DCT) to convert the images from the pixel domain to the frequency domain for high compression efficiency. However, the traditional compression methods cannot be optimized by using large-scale training, which may limit their performance.

Recently, learning based image and video compression methods [6, 7, 10, 11, 25, 28, 30, 33–35, 39, 43–45, 50] attract more and more attention. Ballé et al. [10] propose an end-to-end optimized image compression approach by using the convolutional neural network (CNN) based auto-encoder. To further improve the compression efficiency, Minnen et al. [35] employs the auto-regressive prior information to obtain accurate entropy model and achieve comparable or even better performance than the traditional codec [2].

Although the current learning based methods [28, 35] improve the compression performance, they also increase the computational cost significantly. When compared with the previous learning approaches [10, 11], the current methods [28, 35] exploit the spatial redundancy in the latent feature space by using auto-regressive prior information. Therefore, the decoding procedure in [28, 35] is performed sequentially for each pixel, while the previous approaches [10, 11] can reconstruct all the pixels through convolution layers in a parallel manner. As shown in Table 1, the average GPU decoding time for images with the resolution of $768 \times 512$ using Ballé's method [11] is **0.013 seconds** while the corresponding decoding time using Minnen's method [35] is **2.426 seconds**.

In this paper, we ask the question: **Is it possible to improve the compression efficiency without significantly increasing the computation time?** To address this issue, we propose a unified framework named as Efficient Deep Image Compression (EDIC), which consists of three new components, including the channel attention module, the Gaussian mixture model and the decoder-side enhancement module. Specifically, we utilize an auto-encoder style network for building the image compression framework. To further improve the compression performance, we also exploit the channel relationship in latent features at the encoder side and use an effective channel attention module to enhance the corresponding representation power. More importantly, instead of using the single Gaussian model for entropy estimation like [10, 11, 28, 35], we propose to use Gaussian mixture model (GMM) for more accurate entropy estimation. Besides, we introduce the decoder-side enhancement module to reduce the compression artifacts. The channel attention technique, the Gaussian mixture model and the decoder-side enhancement module are seamlessly combined, which leads to much better image compression performance with only slightly increased computational cost when compared with auto-regressive prior technique in [28, 35]. Experimental results demonstrate that

**Table 1** Decoding Time and BDBR improvement over JPEG2000 [41] of different methods on the Kodak [1] image dataset

| Methods | Decoding Time | BDBR |
|---|---|---|
| Ballé's [11] | 0.013s | 29.87% |
| Minnen's [35] | 2.426s | 53.14% |
| EDIC(Ours) | 0.016s | 53.35% |

The Deoding time is evaluating using one RTX 2080Ti. The full name of BDBR is "Bjontegaard delta bitrate", which refers to the bitrate relative percentage of reduction under the same PSNR

the proposed image compression approach achieves comparable compression performance when compared with the current approach [35], while the decoding speed of our method is over **150** times faster than [35] for images with the resolution of $768 \times 512$. Our method can be readily used for video compression and also achieves promising results for video compression.

The contributions of this paper are summarized in the following aspects. First, to the best of our knowledge, we are the first to introduce the channel attention technique to improve image compression efficiency. Second, the Gaussian mixture model is introduced to model the distribution of the latent representation in a more accurate way. Third, we additionally apply the decoder-side enhancement module to further improve image compression performance. Fourth, our proposed EDIC framework achieves promising image compression performance while significantly reducing the decoding time when compared to Minnen's method [35]. Fifth, the proposed framework is general and also improve the performance of the recent learning based video compression system [33].

Similar ideas were also proposed to boost the image compression performance in two recent works [16, 29]. However, we would like to highlight that our work is a concurrent work as both works [16, 29] as the dates that the three works appeared in arxiv.org are very close to each other. Moreover, our work is different with both works [16, 29] in the following three aspects. First, both works [16, 29] utilize the context information based on the work [35], which are very slow as the method in [35]. In contrast, our work builds upon the method in [11] instead of the algorithm in [35], thus our work is much faster than both works [16, 29]. Second, our proposed attention module is to exploit the channel-wise relationships of the latent representations, while Cheng et al. [16] introduce spatial attention scheme for image compression. Third, we also use our newly proposed technologies (i.e., Gaussian Mixture model, Channel attention scheme and Decoder-side enhancement method) for video compression and achieve promising results on the benchmark datasets, which are not discussed in [16, 29].

## 2 Related work

### 2.1 Traditional image and video compression

The image and video compression techniques are widely used to save the bandwidth and storage size in practical applications. In the past decades, a lot of image and video compression methods have been proposed and several standards are also successfully built. To improve the compression efficiency, the traditional image and video compression methods [2, 3, 41, 46] rely on manually designed techniques, such as liner transform and block based motion estimation and motion compensation schemes.

The image compression methods mainly focus on reducing the spatial redundancy in images. One straightforward method is to convert the images from the pixel domain to the frequency domain, which is easier for compression. For example, the JPEG [46] uses the discrete cosine transform while JPEG2000 [41] employs discrete wavelet transform. After the transform procedure, these coefficients are quantized, and then are sent to the decoder side. To further improve the compression efficiency, the quantized coefficients are losslessly encoded by using the entropy coding tools, such as arithmetic coding [49]. Recently, the intra prediction technique in video compression is also exploited for image compression. For example, the BPG [2] standard is based on HEVC/H.265 [42], which achieves better image

compression performance when compared with the previous image codecs, such as JPEG and JPEG2000. The BPG standard adopts the prediction-transform technique and employs 35 encoding modes to obtain the predicted image, which further reduces the spatial redundancy.

Video compression is used to reduce the temporal redundancy in video sequences. Most video compression algorithms follow the hybrid coding architecture for high compression efficiency. In particular, H.264 [4] is the most widely used video codec. In H.264, the block based motion estimation and motion compensation modules are utilized to obtain the predicted frame. Then we can calculate the residual information, which is compressed by using linear transform. Recently, HEVC/H.265 [42] and versatile video coding (VVC) are proposed as the next generation video codecs. These standards build upon the previous hybrid coding architecture and utilize more advanced techniques for high efficiency coding. For example, HEVC uses the so-called Coding Unit (CU) Tree technique with the CU size ranging from $64 \times 64$ to $8 \times 8$, which provides flexible coding units for different video contents.

### 2.2 Learning based image and video compression

In the past few years, deep neural network (DNN) has demonstrated its effectiveness for a lot of computer vision tasks, including super-resolution, denoising, etc. Recently, researchers try to exploit the powerful representation ability from neural networks to enhance the image/video compression performance [6, 7, 10, 11, 28, 30, 34, 35, 39, 43–45]. Toderici et al. proposed the first learning based image compression framework by using recurrent neural network (RNN). Their approach can generate multiple bitrates through a single model. In [25], more advanced RNN modules and effective reconstruction techniques are introduced to achieve comparable or even better performance when compared with BPG in terms of MS-SSIM [48] However, these methods [25, 44, 45] are designed to minimize the bitrates instead of considering the rate-distortion trade-off.

In [10], Ballé et al. proposed a CNN based image compression framework by optimizing rate-distortion criterion. To improve the accuracy of the entropy model, a hyper-prior model is proposed in [11], where the latent representations are modeled based on zero-mean Gaussian distribution. In [35], Minnen et al. employed the auto-regressive priors to further improve the compression and achieve better performance than BPG in terms of PSNR. However, these CNN based image compression systems have to train different models for different bitrates and increase the model sizes significantly. In [17], Choi et al. proposed a variable rate deep image compression framework by using a conditional autoencoder and generates different bitrates through a single model.

Considering that the quantization procedure itself is not differentiable, it is non-trivial to optimize the image compression system in an end-to-end manner. In [10], the quantization operation is approximated by adding uniform noise in the training stage. In [43], the gradients of quantization operation in the training stage are replaced for end-to-end optimization. To further improve the compression efficiency, Rippel et al. [39] used the multi-scale image decomposition technique to exploit the relationship between different scales. Agustsson et al. [7] proposed a generative adversarial network based image compression system, which provides a visually pleasing reconstructed image for very low bitrate compression. In addition, Li et al. [30] investigated the spatial relation in the latent representations and computed the importance map to guide the learning based image compression method. Inspired by the intra-prediction technique in traditional video coding, Baig et al. used the inpainting method [8] to obtain the predicted block in the reconstructed frame and encode the corresponding residual by using neural network. Although the current learning based methods [28, 35] achieve better

performance than the traditional methods such as BPG [2], the computational cost increases nearly 100 times when the auto-regressive prior [28, 35] is employed. Therefore, it is critical to build a more efficient image compression framework for practical applications.

Recently, learning based video compression has attacted more and more attention. Wu et al. formulated video compression as frame interpolation and applied neural network to encode the residual information. Lu et al. [33] followed the traditional hybrid coding architecture and employed the neural networks to implement the video compression procedure, which can be optimized in an end-to-end manner. Cheng et al. [15] used the interpolation loop in the coding procedure and designed a spatial energy compaction-based penalty term into the loss function for better coding efficiency. In [22], a 3D autoencoder scheme is proposed for video compression without computing the motion information. In [19], the proposed framework can decode the latent representations into motion and blending coefficients. Besides, the residual information is compressed in the latent space instead of the pixel domain. Recently, some related works have been proposed for both the compression and machine learning tasks [13, 20, 24, 52].

# 3 Proposed methods

## 3.1 Overall architecture for image compression

In this section, we introduce the proposed efficient deep image compression framework called EDIC. The architecture of the proposed scheme is illustrated in Fig. 1. Inspired by the recent progress in learning based image compression [10, 11], we also utilize the auto-encoder style network for learning based image compression. Specifically, there are four modules in the proposed scheme, *i.e.,* encoder network, decoder network, hyper-encoder network, and hyper-decoder network. The encoder network takes the original image $x$ as the input and
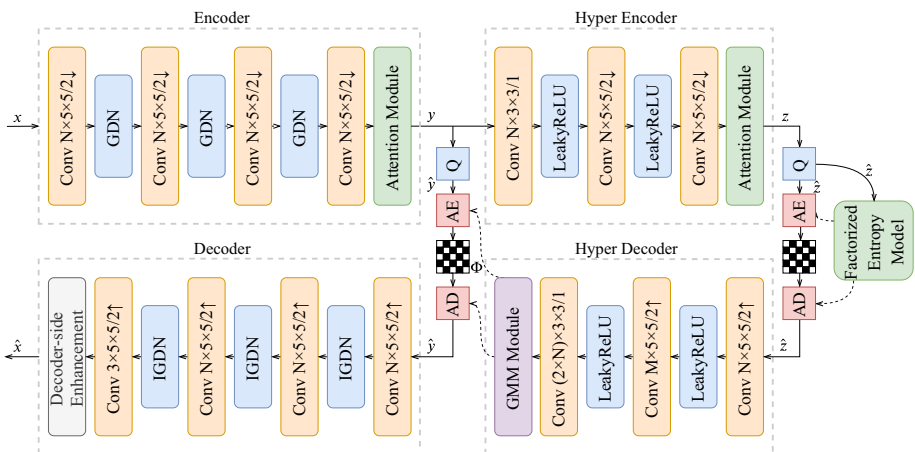


**Fig. 1** The framework of our proposed EDIC. Each convolution layer is denoted by the number of filters, kernel size, and stride. → indicates downsampling, and ← indicates upsampling in each convolutional layer. $N$ and $M$ are the hyper-parameters to set the number of channels for a specific layer."GDN" means generative divisive normalization proposed in [9], and "IGDN" means inverse GDN."Q" denotes quantization. "AE" and "AD" represent arithmetic encoder and arithmetic decoder, respectively."Φ" refers to the estimated parameters of the Gaussian mixture model

generates the corresponding latent representations $y$ by using several convolutional layers and non-linear functions. The latent representations $y$ will be quantized to $\hat{y}$. Following arithmetic coding, like arithmetic encoder and arithmetic decoder, the quantized latent representations $\hat{y}$ were sent to the decoder network to reconstruct the final decoded image $\hat{x}$. We adopt the same quantization strategy as [11, 35]. Considering that the image compression methods aim to achieve high quality reconstructed image at a given bitrate target and entropy model is used to estimate the bitrate, it is critical to build an accurate entropy model. In the proposed framework, we follow the pipeline in [11, 35] and apply the hyper-encoder and hyper-decoder modules to estimate the parameters for the entropy model. Specifically, based on the latent representations $y$, the hyper-encoder module obtains the hyper-prior information and encodes it to latent representations $z$. Similarly, the latent representations $z$ will be quantized as $\hat{z}$. Then, quantized $\hat{z}$ will be sent by arithmetic coding. Finally, the hyper-decoder will reconstruct the hyper-prior information by using quantized hyper-latent representations $\hat{z}$ as the input and estimate the corresponding parameters $\Phi$ of the entropy model. The entropy model of hyper-latent representations is the same as [10, 11]. The network architecture and entropy model in our proposed method will be discussed in the next three sections.

The whole learning based image compression framework is optimized by considering the rate-distortion trade-off in the following way:

$$L = \lambda D + R = \lambda d(x, \hat{x}) + H(\hat{y}) + H(\hat{z}), \tag{1}$$

where $D$ and $R$ represent the distortion and bitrate, respectively. $\lambda$ is the trade-off parameter. $d(\cdot)$ is the distortion metric (mean square error or MS-SSIM [48]). $H$ represents the bitrate for encoding latent representations $\hat{y}$ and $\hat{z}$. In the proposed method, the bitrate is approximated by using the entropy of the corresponding latent representations, *i.e.,* $H(\hat{y}) = E[-log_2(p_{\hat{y}|\hat{z}}(\hat{y} \mid \hat{z}))]$ and $H(\hat{z}) = E[-log_2(p_{\hat{z}}(\hat{z}))]$ $p_{\hat{y}|\hat{z}}(\hat{y} \mid \hat{z})$ and $p_{\hat{z}}(\hat{z})$ represent the distributions of $\hat{y}$ and $\hat{z}$, respectively.

## 3.2 Channel attention scheme

In [28, 35], the auto-regressive prior model which captures the spatial relationship in latent representations is used to improve the compression performance. Meanwhile, some works have applied spatial attention mechanisms implemented by non-local blocks [47] to image compression [14, 32], which aims to reduce the spatial redundancy. Based on the aforementioned two motivations and inspired by [23], we propose to use a light-weight channel attention technique to exploit channel attention in the latent representations $\hat{y}$ and $\hat{z}$. The architecture of the proposed attention module is shown in Fig. 2. Let us denote the input feature map as $\mathbf{X}, \mathbf{X} \in \mathbb{R}^{I \times J \times C}$, where $I$, $J$, and $C$ denote height, width and channel dimension of the feature map, respectively. First, we apply global average pooling to obtain the channel-wise statistics $\mathbf{t} \in \mathbb{R}^C$, which is formulated bellow:

$$t_c = \frac{1}{I \times J} \sum_{i=1}^{I} \sum_{j=1}^{J} x_c(i, j), \tag{2}$$

where $t_c$ means the $c$-th element of $\mathbf{t}$, and $x_c(i, j)$ represents the $c$-th channel specific value of the input feature map $\mathbf{X}$. Then, we apply several non-linear transforms to capture the channel-wise relationship. Specifically, the non-linear transforms are described in the following formula:

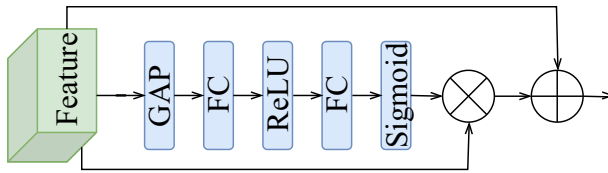$$\mathbf{s} = \sigma(\mathbf{W_2}\delta(\mathbf{W_1 t})), \tag{3}$$

**Fig. 2** The structure of our channel attention module. "GAP" represents global average pooling. "FC" means fully-connected layer

where **s** refers to the output channel-wise attention value, and $\mathbf{W_1} \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W_2} \in \mathbb{R}^{C \times \frac{C}{r}}$ denotes the fully-connected layers, $\delta$ is the ReLU activation function [36] for non-linear transform, and $\sigma$ represents sigmoid activation. For reducing the dimension, we set $r$ as 16. Finally, we re-scale the input feature map $\mathbf{X}$ with **s**. In addition, we add the residual operation in our implementation.

As shown in Fig. 1, the proposed channel attention module is integrated into the encoder network and hyper encoder network and utilized to exploit the channel relationship for high quality compression. We apply the re-weighted feature map to the following quantization and entropy coding modules.

### 3.3 Gaussian mixture model for entropy estimation

In the learning based image compression methods, accurate bit rate estimation is critical. In [28, 35], the learning based systems adopt the hyper-prior compression scheme and the latent representations $\hat{y}$ are modeled as Gaussian distribution as follows:

$$p_{\hat{y}\|\hat{z}}(\hat{y}\|\hat{z}) \sim \mathcal{N}(\mu, \sigma), \tag{4}$$

where $p_{\hat{z}}(\hat{z})$ is represented by using the factorized entropy model [10]. The goal of hyperencoder and hyper-decoder is to estimate the parameters $\mu$ and $\sigma$ of the Gaussian model.

Although the single Gaussian based entropy model has achieved significant improvements when compared with the previous work [10], the representation ability of a single Gaussian model is still limited, especially for complex contents. Therefore, we utilize the Gaussian mixture model to further improve the efficiency of the image compression system. Specifically, the distribution of $\hat{y}$ is formulated as follows:

$$p_{\hat{y}\|\hat{z}}(\hat{y}\|\hat{z}) \sim \sum_{i=1}^{F} \omega_i \mathcal{N}(\mu_i, \sigma_i), \tag{5}$$

where $\omega_i$ represents the weights for different Gaussian models. $F$ is the number of Gaussian models. As shown in Fig. 3, we design three convolutional layers with two LeakyReLU layers to estimate the parameters $\Phi$ of the Gaussian mixture model. In our implementation, $F$ is set as 2. So the output channel number $K$ of the GMM module is set as $5 \times N$, the first $4 \times N$ channels are used to estimate the mean and variance of two Gaussian models, respectively. In order to estimate the weights of each gaussian model. We add a sigmoid layer on the output of the last $N$ channels. If the weight of one Gaussian model is $w$, the weight of another Gaussian model is $(1 - w)$. Specifically, if we design $F(F \geq 3)$ Gaussian models, we can change the number of output channels of the GMM module to $3 \times F \times N (K = 3 \times F)$. Similarly, the first $2 \times F \times N$ channels estimate the mean and variance parameters of $F$ Gaussian models. In particular, we add the softmax layer after the last $F \times N$ channels to calculate the weight of each Gaussian model.

**Fig. 3** The structure of our GMM Module. *N* denotes the hyper-parameter to set the number of channels for a specific layer, and *K* depends on the number of Gaussian models. (See Section 3.3 for more explanations)
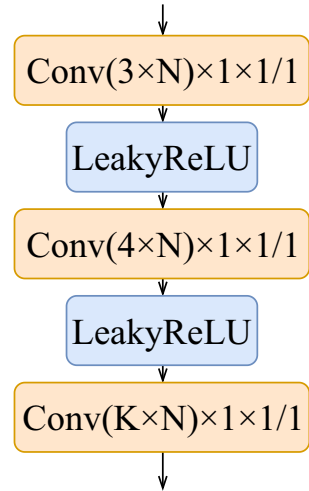


We provide further analysis of the GMM module. As shown in Fig. 4, we visualize the bit allocation map of the latent representations $\hat{y}$. Specifically, the left column is the original image from Kodak. The middle left column is the bit allocation map of $\hat{y}$ after using a single Gaussian model as the entropy model. The middle right column is the bit allocation map of $\hat{y}$ after using the Gaussian mixture model as the entropy model. The right column shows the estimated bit allocation map difference of the latent representations $\hat{y}$ between the single Gaussian model and the Gaussian mixture model. In Fig. 4, the brighter region indicates that the Gaussian mixture model saves more bits, and it is clear that the Gaussian mixture model can save more bits, especially in the edge regions, which further demonstrates the effectiveness of the Gaussian mixture model.

### 3.4 Decoder-side enhancement

Since the proposed compression scheme is a lossy procedure, the reconstructed image has compression artifacts inevitably. To further improve the reconstructed quality, we introduce an enhancement module at the decoder side after image reconstruction. We adopt several residual blocks to restore the original image based on the input reconstructed image. Inspired by the network design strategy for super resolution [31], we introduce the residual block to learn the high frequency information for image compression. As shown in Fig. 5, we first add a convolution layer to increase the channel dimension from 3 to 32. Then, we apply three
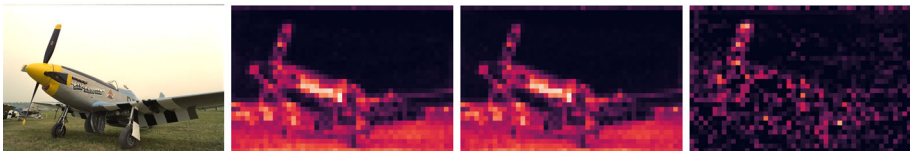


**Fig. 4** Bit allocation map of the latent representations $\hat{y}$. The left column is the original image from Kodak [1]. The middle left column is the bit allocation map of $\hat{y}$ after using a single Gaussian model as the entropy model. The middle right column is the bit allocation map of $\hat{y}$ after using the Gaussian mixture model as the entropy model. The right column is bit allocation difference between them. We take "kodim20.png" for visualization
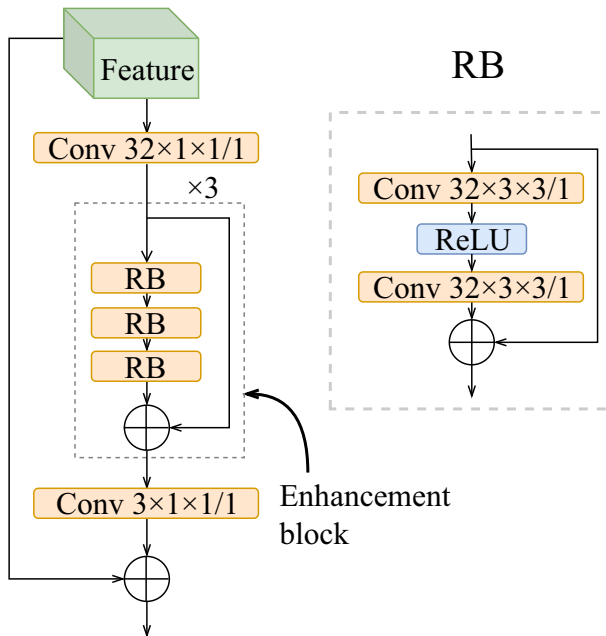
**Fig. 5** The structure of our decoder-side enhancement module. "RB" refers to the residual block

enhancement blocks to the output of the convolution layer. Every enhancement block has three residual blocks. Finally, we add a convolution layer and residual operation to obtain the reconstructed image. Moreover, the decoder-side enhancement module can be readily integrated into the whole compression system and optimized in an end-to-end manner with high efficiency.

### 3.5 Extension for video compression

In order to further demonstrate the effectiveness of our newly proposed method, we also apply our proposed method for the video compression task. In our work, we choose DVC [33] as our baseline algorithm.

The overall framework is shown in Fig. 6. $\{x_1, x_2, ..., x_{t-1}, x_t\}$ denote the current video sequences. $x_t$ refers to the frame at time-step $t$. $\hat{x}_t$ represents the reconstructed frame. $m_t$ and $r_t$ are the motion and residual information, respectively. The procedure of our video compression framework is shown as follows.

### 3.5.1 Motion estimation and compression

We utilize the CNN model proposed by [38] to predict optical flow, which represents the motion information $v_t$. Instead of encoding the motion information $v_t$ directly, we send $v_t$ to the encoder network of the motion compression module to obtain $m_t$, Then, we will quantize $m_t$ and reconstruct the motion information $\hat{v}_t$ by using the decoder network of the motion compression module.
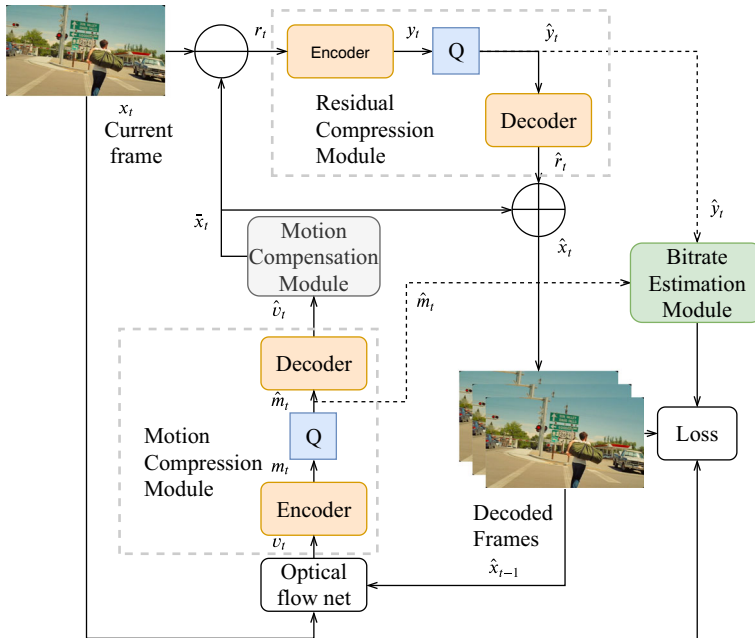
**Fig. 6** The framework of our video compression method. The network structures of the residual compression module and the motion compression module are the same as in the Fig. 1. The bitrate estimation module is our method for estimating the bitrate of the latent representations. "Q" denotes quantization

### 3.5.2 Motion compensation, residual compression and frame reconstruction

The motion compensation module takes the previous reconstructed frame $\hat{x}_t$ and motion information $\hat{v}_t$ as the input, and obtains the predicted frame $\bar{x}_t$, which is supposed to be as close to the current frame $x_t$ as possible. After that, we use the original frame $x_t$ and $\bar{x}_t$ to obtain residual information $r_t$, where $r_t = x_t - \bar{x}_t$. The encoder network of the residual information module encodes the resiudal information $r_t$, and quantizes $r_t$ to obtain the latent representations $y_t$. Similarly, the decoder network of the residual information module reconstructs the residual information $\hat{r}_t$. Then, the final reconstructed frame $\hat{x}_t$ can be obtained, where $\hat{x}_t = \hat{r}_t + \bar{x}_t$.

### 3.5.3 Optimization of the framework

The overall framework is optimized by minimizing the following Rate-Distortion trade-off:

$$L_t = \lambda D_t + R_t = \lambda d(x_t, \hat{x}_t) + H(\hat{r}_t) + H(\hat{m}_t), \tag{6}$$

where $L_t$ is the loss at the current time step $t$, $d(\cdot, \cdot)$ is the distortion between the current frame $x_t$ and the reconstructed frame $\hat{x}_t$, and $H(\hat{r}_t)$ and $H(\hat{m}_t)$ are the bitrates of the latent representations $\hat{r}_t$ of residual information and the latent representations $\hat{m}_t$ of motion information, which are estimated by the bitrate estimation module.

DVC utilizes the method proposed by Ballé [11] to compress the residual information, and Ballé's method [10] to compress the motion information. In our work, we propose to use our proposed EDIC image compression framework to compress both the residual and motion

information. Specifically, in the encoder network of the residual compression module and the motion compression module, we utilize the proposed channel attention scheme described in Section 3.2 to reduce the redundancy of the latent representations of residual and motion information. In terms of bitrate estimation module, we introduce the newly proposed Gaussian mixture model as the entropy model described in Section 3.3 to estimate the bitrates of the latent representations more accurately, in which the hyper-encoder and hyper-decoder network are used to estimate the parameters of Gaussian mixture model. Furthermore, in the decoder network of the residual compression module and the motion compression module, we add the decoder-side enhancement module in Section 3.4 to improve the reconstructed qualities of the residual and motion information effectively.

## 4 Discussion

**Differences and dependencies of the three modules.** (1). Differences. Firstly, the channel attention module is employed to exploit the latent representations' channel relationship. Secondly, the Gaussian mixture model is introduced specifically for the entropy model, enhancing the accuracy of bitrate estimation. Lastly, the decoder-side enhancement module is utilized to improve both the reconstructed quality and overall image compression performance. (2). Dependencies. It is worth mentioning that these three modules are proposed independently, with each module serving a unique purpose.

**The advantages of EDIC.** In this paper, we enhance image compression efficiency by incorporating the channel attention technique. Additionally, we improve the accuracy of modeling the latent representation distribution through the Gaussian mixture model. Moreover, we implement the decoder-side enhancement module to further optimize the image quality and improve the image compression performance. By integrating these three modules, our proposed EDIC framework achieves superior image compression performance while significantly reducing decoding time compared to Minnen's method (see Table 1 of the main paper). Furthermore, our framework is applicable to a wide range of scenarios and exhibits improved performance in the field of learning-based video compression.

## 5 Experiments

In this section, we perform extensive experiments to demonstrate the effectiveness of our proposed EDIC framework, which consists of the attention module, the GMM module and the decoder-side enhancement module. With regard to image compression, we adopt 20745 high-quality images from Flick.com

and randomly take $256 \times 256$ cropped patches for training. For performance evaluation, we calculate the Rate-Distortion (RD) performance, which is averaged over all images in the Kodak PhotoCD image dataset [1]. For video compression, we use Vimeo-90k [51] dataset, which has 89,800 video clips with the resolution of $256 \times 256$, as our training dataset, and evaluate our model on the HEVC Standard Test Sequences (*i.e.*, Class B, Class C, Class D, Class E) [42], which is widely used for evaluating video compression methods. Our EDIC framework is implemented on the PyTorch [37] platform. All the experiments are conducted on the GPU NVIDIA 2080Ti server with 11 GB memory.
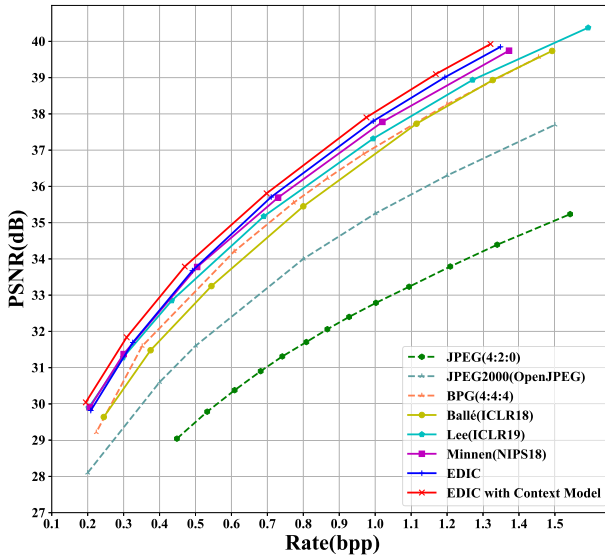
**Fig. 7** Rate-distortion curves of our proposed EDIC method and the competitive methods for image compression when using the PSNR metric. The "Context Model" is from Minnen's work [35], which must be executed sequentially in the inference stage

## 5.1 Performance and implementation details for image compression

For image compression with the quality metric as the MSE loss function, we train our model using different $\lambda$ values (*i.e.*, 256, 512, 1024, 2048, 4096, 6144 8192). In the first stage, we train the high bitrate point in the Rate-Distortion (RD) curve with $\lambda$ as 8192. The model is trained on 1 GPU with the batch size of 4. We apply Adam optimizer [27] with the learning rate of $1 \times 10^{-4}$ in the first 3,000,000 iterations and $1 \times 10^{-5}$ in the remaining 500,000 iterations. For other bitrates, we just adopt the model trained on high bitrate ($\lambda = 8192$) as a pre-trained model and fine-tune our model. We use Adam optimizer with the learning rate of $1 \times 10^{-4}$ in the first 500,000 iterations, and $1 \times 10^{-5}$ in the remaining 500,000 iterations. Other training settings remain the same. When our model is optimized with other quality metrics, such as the MS-SSIM loss function, we adopt the model optimized by the MSE loss function with $\lambda$ of 8192 as our pre-trained model. Then, we change the MSE loss function to the MS-SSIM loss function and fine-tune the pre-trained model with different $\lambda$ values (*i.e.*, 16, 32, 64, 128, 256, 384, 512). We train the model with the learning rate of $1 \times 10^{-5}$ for 500,000 iterations. Besides, we set $N$ to 320 and $M$ to 480.

As shown in Fig. 7, we adopt peak signal-to-noise ratio (PSNR) as the quality metric. We compare our EDIC method with the well-kown image compression standards, like BPG [2], JPEG [46], JPEG2000 [41], and recent neural networks methods, like Ballé's work [11], Minnen's work [35] and Lee's work [28]. The results of Lee's work [28] are from their released source code[1]. The results of Ballé's work [11] and Minnen's work [11] are based on our implementation. When compared with the traditional methods, our EDIC has surpassed BPG [2], JPEG [46], JPEG2000 [41] by a large margin. When compared with the existing deep learning based methods, our EDIC achieves significant improvement over Ballé's work [11].

---

[1] https://github.com/JooyoungLeeETRI/CA_Entropy_Model

As far as we know, the method proposed by Minnen et al. has achieved promising performance for image compression. Our method has comparable results with Minnen's work [35] and Lee's method [28] at low bitrates, and achieves apparent performance improvement over Minnen's work [35] and Lee's method [28] at high bitrates. In addition, Minnen's work and Lee's method are very slow, because their inference strategies are sequential. By contrast, our method can be readily parallelized. As a result, our method is very efficient, which is very important for practical application scenarios. Furthermore, the attention module, the GMM module, and the Decoder-side Enhancement module are all independent modules and can be easily incorporated with other methods. As shown in Fig. 7, when we incorporate our method into Minnen's work [35], which has the context model for estimating more accurate entropy parameters, our EDIC method with context model also achieves over 0.2 dB improvement when compared with our EDIC method. which again demonstrated the effectiveness of our proposed schemes.

As shown in Fig. 8, we also conduct the experiments in terms of the MS-SSIM quality metric. In order to describe the improvement more clearly, we report the MS-SSIM values using decibels (*i.e.*, $-10\log_{10}(1-\text{MS-SSIM})$). It is clear that our EDIC is better than BPG [2], JPEG [46], JPEG2000 [41], and Ballé's [11]. When compared with the other methods, our EDIC is comparable with Minnen's method [35] and lower than Lee's work [28] at low bitrates. However, our EDIC is apparently superior to their methods at high bitrates.

## 5.2 Performance and implementation details for video compression

For the implementation details on video compression, we follow the settings of DVC [33]. Each video clip in the Vimeo dataset consists of 7 frames. The HEVC test dataset contains the videos with different resolutions and different contents. We set $N$ to 192 and $M$ to 288 in Fig. 1. Other modules of our video compression framework is same as DVC [33]. In the
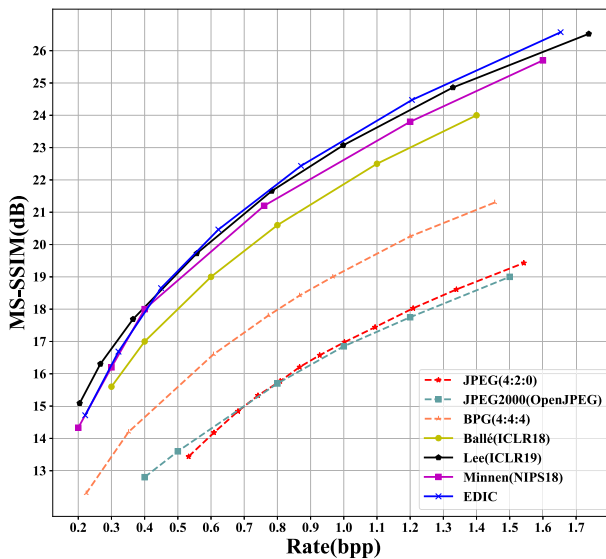


**Fig. 8** Rate-distortion curves of our proposed EDIC method and the competitive methods for image compression when using the MS-SSIM metric

training process, when using the quality metric as the MSE loss function, we select $\lambda = 4096$ to obtain our pre-trained model for 2,000,000 iterations with the learning rate of $1 \times 10^{-4}$. Then, we apply different $\lambda$ values (*i.e.*, 256, 512, 1024, 2048) to fine-tune this pre-trained model with the learning rate of $1 \times 10^{-5}$. When optimized by using the MS-SSIM loss function, we fine-tune the model at high bitrates from the MSE loss function for 80,000 iterations with the learning rate of $1 \times 10^{-5}$. The remaining training strategies are similar to the implementation details of image compression described in Section 5.1.

As shown in Fig. 9 and Fig. 11, we compare our method with the traditional video compression standards, like H.264 [4], H.265 [5], and the deep learning based method DVC [33]. To obtain the compressed frames by the H.264 and H.265, we apply the FFmpeg with very fast mode, and set the GOP sizes of the HEVC dataset to 10. As for the Rate-Distortion(RD) curves, in terms of the PSNR quality metric, our method is much better than DVC [33] and H.264 [4], and it achieves comparable performance with H.265 [5]. With regard to MS-SSIM, it is clear that our newly proposed method is superior to DVC [33], H.264 [4], and H.265 [5] for almost all the HEVC test classes.

## 5.3 Ablation study

### 5.3.1 Effectiveness of each module

In order to verify the effectiveness of each proposed module, we perform ablation study for image compression in this section. For the baseline model, we utilize a single Gaussian model as our entropy model. When implementing the baseline model, we just remove the GMM module, the attention module and the decoder-side enhancement module (See Fig. 1). After that, the last convolution layer of the hyper decoder is $2 * N$, so the first $N$ channels are used to estimate the mean parameters and the last $N$ channels are used to estimate the variance parameters of a single Gaussian model. Then, we add each module to the baseline model, respectively. When we utilize the Gaussian mixture model as our entropy model, we add the GMM module described in Fig. 4 based on the baseline model to estimate the parameters of the Gaussian mixture model. As shown in Fig. 10, we compare the performance of the
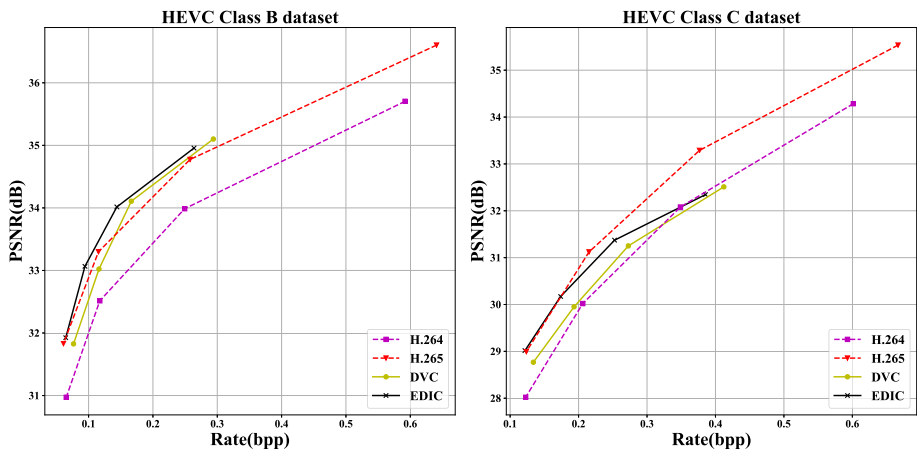


**Fig. 9** Rate-distortion curves of our proposed EDIC method and the competitive methods for video compression when using the PSNR metric
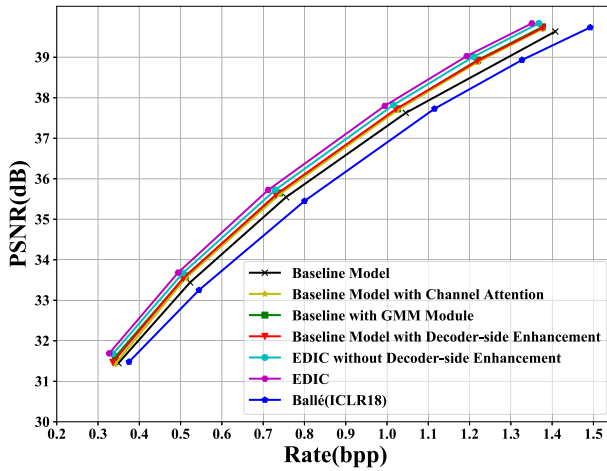
**Fig. 10** Effectiveness of each module in our newly proposed framework

baseline model, the baseline model with an additional attention module, the baseline model with an additional decoder-side enhancement module, the baseline model with additional GMM module, our proposed EDIC method without decoder-side enhancement module, and our overall EDIC method consisting of the GMM module, the attention module and the decoder-side enhancement module. For all experiments, we use the same training strategy described in Section 5.1. As shown in Fig. 10, we observe that each module brings significant performance improvement when compared to our baseline model. For the attention module, the baseline model with the attention module is about 0.2 dB better than to the baseline model. The baseline model with the GMM module is also superior to the baseline model, which demonstrates the effectiveness of the Gaussian mixture model. Furthermore, when we add the decoder-side enhancement module to any models, we can achieve better performance (Fig. 11).
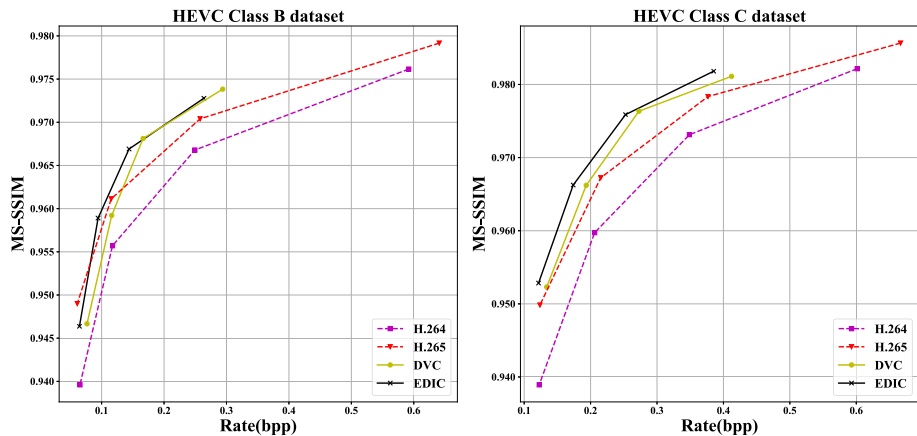


**Fig. 11** Rate-distortion curves of our proposed EDIC method and the competitive methods for video compression when using the MS-SSIM metric
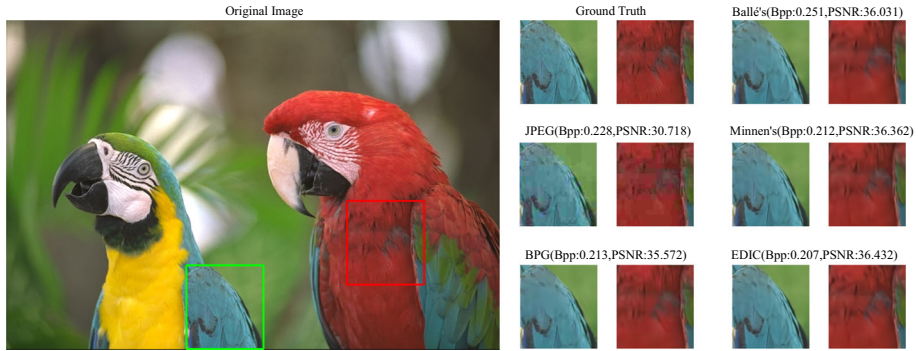
**Fig. 12** Visualization of reconstructed sample images of ground truth, JPEG [46], BPG [2], Ballé's method [11], Minnen's method [35] and our proposed EDIC method. We take "kodim23.png" from Kodak [1] for illustration

## 5.4 Visualization

In order to demonstrate the effectiveness of our EDIC more clearly, we provide some visualization results. We compare the reconstructed sample images of our proposed EDIC method and other competitive methods in Fig. 12. The results of the learned image compression methods are optimized by MSE loss function. The reconstructed image of our method achieves higher quality in both PSNR metric and qualitative viewing when the compression ratios of all methods are close.

## 5.5 Extension on compressed images on face recognition

**Performance on face recognition** We have also conducted a comparison on the face recognition task by using MobileNetV2 [40] backbone network. All experiments are established based on the criterion of ArcFace [18] with $m = 0.5$. Based on the above settings, we implement experiments on training datasets of MS-Celeb-1M [21] and VGG2-Face [12], and evaluate the performance on MegaFace [26]. Specifically, in Table 2, we compressed the testing MegaFace dataset by using different coding strategies. When compared with other coding methods, our coding strategy achieves the best results at $10^5$ and $10^6$ distractors.

## 6 Conclusion

In this paper, we have proposed a unified framework EDIC to boost image compression performance while keeping fast inference speed for practical scenarios. We first adopt a

**Table 2** Rank-1 accuracy with different distractors on MegaFace

| Method | Distractors | |
| --- | --- | --- |
| | $10^5$ ↑ | $10^6$ ↑ |
| Baseline (Upper Bound) | 94.56 | 90.25 |
| Ballé [11] | 93.82 | 89.50 |
| Minnen's [35] | 94.07 | 89.63 |
| Ours | **94.46** | **90.05** |

lightweight channel-wise attention mechanism to reduce channel-wise redundancy of the latent representations. Moreover, we propose to use the Gaussian mixture model to estimate the bitrate more accurately, which has been shown to be very useful for edge regions. Finally, we introduce a simple decoder-side enhancement module to further improve image compression performance. Our framework can be trained in an end-to-end fashion and readily used for video compression. Experimental results have demonstrated the superiority of our proposed EDIC method for image and video compression over the existing methods.

**Data Availability** The datasets generated during and/or analysed during the current study are available online.

## Declarations

**Conflicts of interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

1. kodak E (2018)kodak lossless true color image suite (photocd pcd0992). http://r0k.us/graphics/kodak/
2. bellard F (2018) bpg image format. http://bellard.org/bpg/, accessed: 30 Oct 2018
3. Webp (2018) . https://developers.google.com/speed/webp/, 30 Oct 2018
4. x264 (2018a) the best h.264/avc encoder. https://www.videolan.org/developers/x264.html, 30 Oct 2018
5. x265 (2018b) hevc encoder / h.265 video codec. http://x265.org, 30 Oct 2018
6. Agustsson E, Mentzer F, Tschannen M, et al (2017) Soft-to-hard vector quantization for end-to-end learning compressible representations. In: NIPS, pp 1141–1151
7. Agustsson E, Tschannen M, Mentzer F, et al (2018) Generative adversarial networks for extreme learned image compression. arXiv:1804.02958
8. Baig MH, Koltun V, Torresani L (2017) Learning to inpaint for image compression. In: NIPS, pp 1246–1255
9. Ballé J, Laparra V, Simoncelli EP (2015) Density modeling of images using a generalized normalization transformation. arXiv:1511.06281
10. Ballé J, Laparra V, Simoncelli EP (2017) End-to-end optimized image compression. In: 5th International conference on learning representations, ICLR
11. Ballé J, Minnen D, Singh S, et al (2018) Variational image compression with a scale hyperprior. In: 6th International conference on learning representations, ICLR
12. Cao Q, Shen L, Xie W, et al (2018) Vggface2: A dataset for recognising faces across pose and age. In: IEEE International conference on automatic face & gesture recognition. IEEE, pp 67–74
13. Chamain LD, Racapé F, Bégaint J, et al (2021) End-to-end optimized image compression for machines, a study. In: 2021 Data compression conference (DCC). IEEE, pp 163–172
14. Chen T, Liu H, Ma Z, et al (2019) Neural image compression via non-local attention optimization and improved context modeling. arXiv:1910.06244
15. Cheng Z, Sun H, Takeuchi M, et al (2019) Learning image and video compression through spatial-temporal energy compaction. In: Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR, pp 10,071–10,080
16. Cheng Z, Sun H, Takeuchi M, et al (2020) Learned image compression with discretized gaussian mixture likelihoods and attention modules. arXiv:2001.01568
17. Choi Y, El-Khamy M, Lee J (2019) Variable rate deep image compression with a conditional autoencoder. In: Proceedings of the IEEE international conference on computer vision, pp 3146–3154
18. Deng J, Guo J, Xue N, et al (2019) Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4690–4699
19. Djelouah A, Campos J, Schaub-Meyer S, et al (2019) Neural inter-frame compression for video coding. In: Proceedings of the IEEE International conference on computer vision, pp 6421–6429
20. Duan L, Liu J, Yang W, et al (2020) Video coding for machines: A paradigm of collaborative compression and intelligent analytics. Trans Img Proc 29:8680–8695
21. Guo Y, Zhang L, Hu Y, et al (2016) Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: European conference on computer vision, pp 87–102

22. Habibian A, Rozendaal Tv, Tomczak JM, et al (2019) Video compression with rate-distortion autoencoders. In: Proceedings of the IEEE international conference on computer vision, pp 7033–7042

23. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141

24. Hu Y, Yang S, Yang W, et al (2020) Towards coding for human and machine vision: A scalable image coding approach. In: 2020 IEEE international conference on multimedia and expo (ICME). IEEE, pp 1–6

25. Johnston N, Vincent D, Minnen D, et al (2018) Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In: CVPR

26. Kemelmacher-Shlizerman I, Seitz SM, Miller D, et al (2016) The megaface benchmark: 1 million faces for recognition at scale. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4873–4882

27. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980

28. Lee J, Cho S, Beack SK (2018) Context-adaptive entropy model for end-to-end optimized image compression. arXiv:1809.10452

29. Lee J, Cho S, Kim M (2019) A hybrid architecture of jointly learning image compression and quality enhancement with improved entropy minimization. arXiv:1912.12817

30. Li M, Zuo W, Gu S, et al (2018) Learning convolutional networks for content-weighted image compression. In: CVPR

31. Lim B, Son S, Kim H, et al (2017) Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 136–144

32. Liu H, Chen T, Guo P, et al (2019) Non-local attention optimized deep image compression. arXiv:1904.09757

33. Lu G, Ouyang W, Xu D, et al (2019) DVC: An end-to-end deep video compression framework. In: Proceedings of the IEEE conference on computer vision and pattern recognition,CVPR, pp 11,006–11,015

34. Mentzer F, Agustsson E, Tschannen M, et al (2018) Conditional probability models for deep image compression. In: CVPR, 2, p 3

35. Minnen D, Ballé J, Toderici GD (2018) Joint autoregressive and hierarchical priors for learned image compression. In: Advances in neural information processing systems, pp 10,771–10,780

36. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814

37. Paszke A, Gross S, Massa F, et al (2019) Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32. Curran Associates, Inc., p 8024–8035

38. Ranjan A, Black MJ (2017) Optical flow estimation using a spatial pyramid network. In: The IEEE Conference on computer vision and pattern recognition (CVPR)

39. Rippel O, Bourdev L (2017) Real-time adaptive image compression. In: ICML

40. Sandler M, Howard A, Zhu M, et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 4510–4520

41. Skodras A, Christopoulos C, Ebrahimi T (2001) The jpeg 2000 still image compression standard. IEEE Signal Process Mag 18(5):36–58

42. Sullivan GJ, Ohm JR, Han WJ et al (2012) Overview of the high efficiency video coding(hevc) standard. TCSVT 22(12):1649–1668

43. Theis L, Shi W, Cunningham A, et al (2017) Lossy image compression with compressive autoencoders. In: 5th International conference on learning representations, ICLR

44. Toderici G, O'Malley SM, Hwang SJ, et al (2016) Variable rate image compression with recurrent neural networks. In: 4th International conference on learning representations, ICLR

45. Toderici G, Vincent D, Johnston N, et al (2017) Full resolution image compression with recurrent neural networks. In: CVPR, pp 5435–5443

46. Wallace GK (1992) The jpeg still picture compression standard. IEEE Transactions on Consumer Electronics 38(1):xviii–xxxiv

47. Wang X, Girshick R, Gupta A, et al (2018) Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7794–7803

48. Wang Z, Simoncelli E, Bovik A, et al (2003) Multi-scale structural similarity for image quality assessment. In: ASILOMAR CONFERENCE ON SIGNALS SYSTEMS AND COMPUTERS, IEEE; 1998, pp 1398–1402

49. Witten IH, Neal RM, Cleary JG (1987) Arithmetic coding for data compression. Communications of the ACM 30(6):520–540

50. Wu CY, Singhal N, Krahenbuhl P (2018) Video compression through image interpolation. In: ECCV

51. Xue T, Chen B, Wu J et al (2019) Video enhancement with task-oriented flow. International Journal of Computer Vision, IJCV 127(8):1106–1125

52. Yang F, Wang Y, Herranz L et al (2022) A novel framework for image-to-image translation and image compression. Neurocomputing 508:58–70