Dec-Adapter: Exploring Efficient Decoder-Side Adapter for Bridging Screen Content and Natural Image Compression

Sheng Shen, Huanjing Yue*, Jingyu Yang*
Tianjin University
{codyshens, huanjing.yue, yjy}@tju.edu.cn

Abstract

Natural image compression has been greatly improved in the deep learning era. However, the compression performance will be heavily degraded if the pretrained encoder is directly applied on screen content image compression. Meanwhile, we observe that parameter-efficient transfer learning (PETL) methods have shown great adaptation ability in high-level vision tasks. Therefore, we propose a Dec-Adapter, a pioneering entropy-efficient transfer learning module for the decoder to bridge natural image and screen content compression. The adapter's parameters are learned during encoding and transmitted to the decoder for image-adaptive decoding. Our Dec-Adapter is lightweight, domain-transferable, and architecture-agnostic with generalized performance in bridging the two domains. Experiments demonstrate that our method outperforms all existing methods by a large margin in terms of BD-rate performance on screen content image compression. Specifically, our method achieves over 2 dB gain compared with the baseline when transferred to screen content image compression.

1. Introduction

With the growing demand for high-resolution images and videos online, efficient and more versatile image compression methods are essential for storage and transmission. In recent years, learning-based image compression (LIC) approaches [2, 3, 29, 8, 47] have been emerging and quickly surpassed traditional codecs, such as JPEG [36], BPG [4] and VVC [11]. These methods learn optimal non-linear transforms and probabilities for entropy coding by training end-to-end networks.

However, the model that minimizes the expected rate and distortion (RD) cost on a specified dataset may not be ideal for every test case due to the difference between the properties of a specific image and the statistic properties of a dataset. It is especially challenging when the test image comes from a different domain. For example, applying a model trained on natural images to screen content images will lead to worse performance. An intuitive approach to solve this problem is fine-tuning the model on a screen content dataset. However, this approach has two drawbacks: it will reduce the model's generalization to its original data domain and consume considerable resources for fine-tuning a large model. Therefore, it is essential to explore alternative solutions to address these problems.

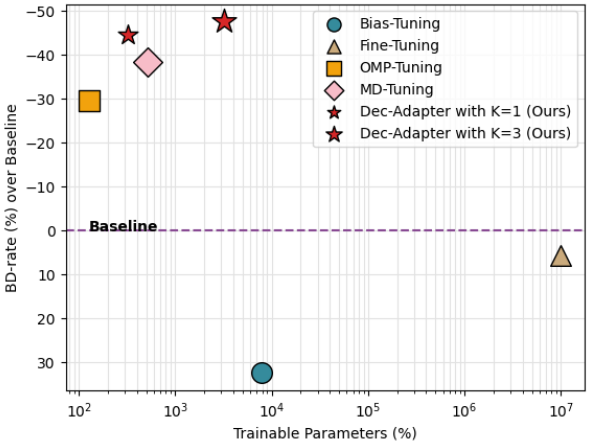


Figure 1. Comparison of our Dec-Adapter against other transfer learning methods on screen content dataset SIQAD [39] in terms of BD-rate. All the methods are based on the same LIC compression model, i.e., Cheng2020 [8] (namely the baseline).

There are some works exploring content-adaptive image compression. One approach is adjusting the parameters of the encoder neural network [35], but this has shown limited improvement, as shown in Figure 1. Another approach is to adapt the the parameters of the decoder [23, 34, 46], which needs to transmit the adapted parameters to the decoder side. Therefore, a large adapter will cause a heavy bitrate overhead. The third approach is to modify the la-

*Corresponding authors. This work was supported in part by the National Natural Science Foundation of China under Grant 62231018 and Grant 62072331.

tent tensor produced by the encoder[24, 13]. Although this process introduces zero parameter, its adaptation ability is limited. The main reason is that it assumes the hyper latent representation follows a Gaussian distribution, which is required for the bit-back [41] coding process.

Meanwhile, we observe that the Parameter-Efficient Transfer Learning (PETL) is utilized in vision and language tasks [17, 25, 40, 7]. It makes a pre-trained model on a large-scale dataset adapt to a new task by fine-tuning a small number of parameters, which is more efficient than fine-tuning the whole network. Different from it, image compression is a self-encoding and decoding structure. Therefore, a more effective fine-tuning method is fine-tuning for each image other than a new dataset. From this perspective, fine-tuning based cross domain image compression task is actually a one-shot task. The training and testing processes only utilize one image, which can avoid the domain shift between training and testing. In addition, different from PETL, the adapter for image compression should be entropy-efficient since the bitrate overhead depends on the entropy of the adapter.

Based on the above observations, we propose an entropy-efficient adapter to bridge screen content and natural image compression. At the encoding side, we adopt a latent refining strategy to enhance the model's adaptability. At the decoding side, instead of updating all parameters, we propose to insert a lightweight adapter into the decoder. The entropy of the adapter is optimized during the encoding process and the adapter parameters will be transmitted to the decoder for image adaptive decoding. The main contributions of this work is summarized as follows.

- To the best of our knowledge, we are the first to systematically investigate feasible solutions for entropy-efficient transfer learning (EETL) for screen content image compression.
- We propose an efficient decoder-side adapter for image adaptive compression. We give a comprehensive analysis on the design of the adapter structure, the decoder structure, and the insertion position.
- Experiments demonstrate that our Dec-Adapter achieves the best BD rate on both the natural image (in-domain) compression and screen content image (out-domain) compression. Compared with the baseline, our approach achieves an improvement of over 2 dB when transferred to the images with a large domain gap.

2. Related Work

2.1. Learned Image Compression

Learned image compression (LIC) optimizes modules in an end-to-end manner for image compression [31]. Most studies aim to reduce distortion by designing more efficient architectures and predictive models, such as encoders and decoders [8, 47, 42] and entropy models [3, 2, 30, 15]. Some studies focus on human perception rather than distortion for image compression [27]. Others explore variable rate compression [9]. Although learned image compression has surpassed conventional codecs on natural images, its performance on screen content image compression has not been well explored. This is because most studies only train and evaluate on natural images (such as [12], Kodak [22]). Therefore, our work investigates adapting the natural image codec to screen contents by introducing a decoder adapter.

2.2. Content-Adaptive Compression

Our work is most related to content-adaptive compression. In learned image compression, content-adaptive compression is achieved by refining the latent representation obtained by the encoder [21, 26, 41, 13, 33, 1, 38, 45, 14] for each image. The latent bitstream can be obtained by compressing the refined latent representation. The parameter update and compression in the decoder side are mainly used for multi-image compression, video compression post-processing and neural video compression [35, 20, 23, 10, 24]. Rozendaal *et al*. [35] update all the parameters in the decoder and entropy model via rate-distortion optimization. However, although they handle multiple images adaptively, this method requires relatively more bits to compress a single image. Lam *et al*. [23] only update the biases in convolutional layers in the post-processing network. Zou *et al*. [46] insert multiplicative parameters (used to multiply the output of convolutional layers) that can overfit and update these parameters for intra-frame coding. However, these updated parameters are selected temporarily and only optimized in terms of distortion. Tsubota *et al*. [34] introduce a matrix as an adapter for decoding and achieve state-of-the-art performance on artist domains. Their work demonstrates the effectiveness of adapters in decoding. However, they do not explore effective ways to use adapters for the decoder and only implement their adapter as matrix multiplication. In contrast, we propose an entropy efficient adapter, which follows a convolutional bottleneck structure with depthwise and pointwise convolution.

2.3. Screen Content Image Compression

Despite the progress in LIC techniques, their adaptation for SCIs is still a relatively less explored domain. Recently, Wang *et al.* [37] proposed a learned image codec that incorporated the concept of transform skip into the end-to-end pipeline to improve SCI compression. They verified that retraining an end-to-end learned pipeline on SCI would improve SCI compression performance. However, the retrained model cannot work well for natural image compression. Afterwards, some works [43, 44] have advocated

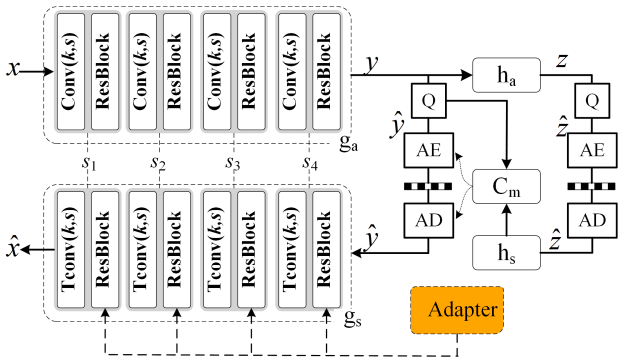


Figure 2. A prominent VAE structure based LIC with hyperpriors, which consists of main and hyper encoder-decoder pairs. The main coders employ 4-stage paired blocks. We use Resblock [16] for illustration as it is widely in image compression [8]. The hyper coders (h_a, h_s) have similar blocks which we omit for simplicity. Convolution Conv (k, s) and its transpose Tconv (k, s) utilize a kernel of size $k \times k$ with a stride of s. $k = 3$ or 5 and $s = 2$. We insert adapters into the decoder to make the pretrained network adapt to different image compression. Note that our adapter can be inserted into any kind of blocks and ResBlock is only an example.

for a segmentation approach that discerns natural regions and synthetic contents, and utilize two different networks to process the two kinds of contents. Different from them, we utilize one network to deal with both natural and screen contents via image based adaptation.

2.4. Parameter-Efficient Transfer Learning

Parameter-efficient transfer learning (PETL) [17, 18, 25, 7] has attracted increasing research interest due to the rapid growth of model size. PETL was first proposed in natural language processing (NLP), where it was used for large-scale pre-trained models. It was verified that almost full fine-tuning performance can be achieved by fine-tuning only a few lightweight modules in large-scale pre-trained models. Based on the success of NLP [17], the PETL strategy was also applied to large-scale pre-trained visual models for various visual tasks. Among them, adapter-based [7, 25] and prompt-based fine-tuning methods [19] are two main paradigms for large-scale visual models. The principle of our proposed decoder adapter is close to previous visual adapter and the main difference is that our dec-adapter should be entropy-efficient. Therefore, our adapter cannot be embedded into every submodule of the model, which is different from the transfer learning in high-level vision tasks.

3. Method

3.1. Preliminary: General Framework for Deep Image Compression

Most of the LIC models share the Variational Autoencoder (VAE) architecture, as shown in Figure 2. The nonlinear transforms g_a, g_s, h_a, and h_s are parameterized by neural networks. The original image is denoted by x. The latent representations before and after quantization are represented by $y = g_a(x)$ and $\hat{y} = Q(y)$, respectively. The hyper latent before and after quantization are denoted by $z = h_a(y)$ and $\hat{z} = Q(z)$, respectively. The side information used to estimate the scale parameter σ for the entropy model of latent $h_a(y)$ is given by $\hat{z}$. During training, the quantization operation is approximated by adding uniform noise to produce differentiable variables $\tilde{y}$, $\tilde{z}$, and $\tilde{x}$. The tradeoff between rate and distortion or the loss function can be expressed as follows:

$$L_{RD} = \underbrace{\mathbb{E}_{x\sim p_x}[-\log_2 p_{\hat{y}|\hat{z}}(\hat{y}|\hat{z}) - log_2 p_{\hat{z}}(\hat{z})]}_{rate} + \underbrace{\lambda \mathbb{E}_{x\sim p_x} D(\hat{x}, x)}_{distortion}. \quad (1)$$

In the following, we introduce our dec-adapter based on the recent state-of-the-art LIC models.

3.2. Overview of Dec-Adapter

As mentioned previously, it is inevitable to encounter the problem of performance degradation when compressing out-of-domain images. Therefore, in this work, we propose to learn an entropy-efficient Dec-Adapters to align the image domain shift while freezing the original decoder model at the receiver side. As shown in Figure 2, our adapter is inserted into the ResBlocks of the decoder side. Figure 3 illustrates different adapting scheme for ResBlocks, which will be detailed in the following.

3.3. Architecture of Dec-Adapter

We utilize a simple and lightweight adapter. The detailed architecture of our adapter is presented in Figure 3 (b). Our adapter follows a bottleneck structure, which is also widely used by PETL methods in NLP tasks [17] and high-level vision tasks [25, 40, 7]. The architecture consists of two convolutional layers cascaded by a GELU activation function. The first convolution squeezes the channel dimension, while the second convolution expands it back. Considering that we need to reduce the entropy of adapter parameters, we first reduce the parameter number. We use a depth-wise convolution with weight $W_{squeeze} \in \mathbb{R}^{\frac{C_{in}}{\alpha} \times \alpha \times K \times K}$ for the first convolution and a point-wise convolution with weight $W_{expand} \in \mathbb{R}^{C_{out} \times \frac{C_{in}}{\alpha} \times 1 \times 1}$ for the second convolution.

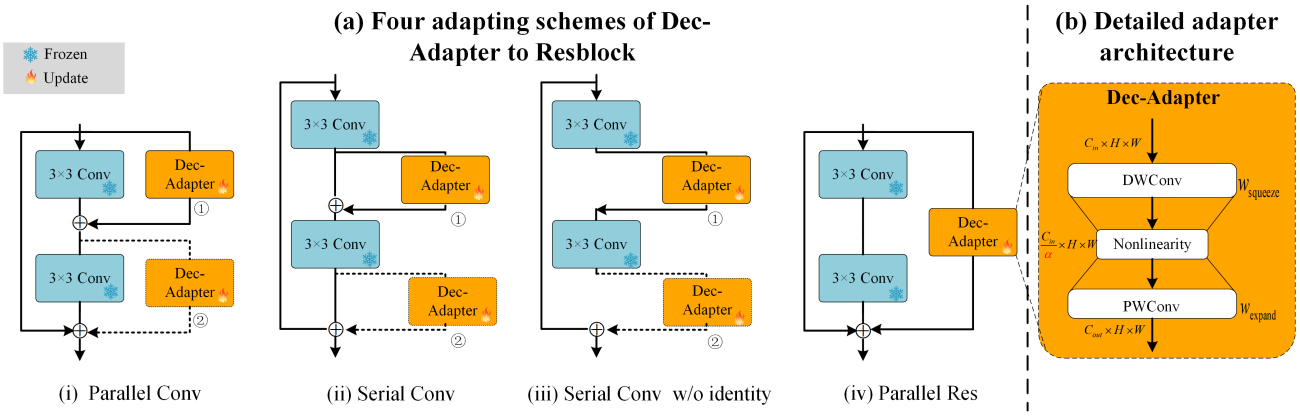


Figure 3. Illustration of (a) four options for applying Dec-Adapter to one ResBlock, and (b) the architecture of Dec-Adapter. Orange color refers to trainable parameters and blue color refers to frozen ones. The adapter can be attached to either ① or ② in (a) (i), (ii) and (iii). We omit the active function in the ResBlock for brevity.

We use a compression factor of α to represent channel dimension squeezing, where α is a hyperparameter discussed in Section 4.2. The adapter computes $\Delta f \in \mathbb{R}^{C_{out} \times H \times W}$ as follows:

$$\Delta f = (W_{up} \dot{\otimes} \sigma(W_{down} \hat{\otimes} f_{in})), \tag{2}$$

where $\dot{\otimes}$ and $\hat{\otimes}$ denotes point-wise and depth-wise convolution, respectively. Finally, the adaption module learns Δf and updates the input latent feature f as:

$$f \leftarrow f + \Delta f \tag{3}$$

The ablation study on different design choices is presented in Section 4.2.

3.4. Adapting Decoder with Dec-Adapter

Since most LIC methods contain ResBlocks at the decoder side, we propose applying adapters to tune a ResBlock in the decoder in four different adapting schemes, as shown in Figure 3 (a). We design these schemes from two perspectives: "where" and "how". The "where" perspective refers to which intermediate feature f in the pre-trained ResBlock is adapted by the adapter. The "how" perspective refers to how the adapter computes the Δf that modifies the feature f. Since a common ResBlock contains two convolution layers, we consider plugging adapters to each convolution layer or the whole residual layer. Moreover, we consider two connection forms of adapters: parallel and serial. Combining these two design strategies, we obtain four variants of adapting schemes with adapters: **Parallel Convolution, Serial Convolution, Serial Convolution w/o identity, and Parallel Residual**. The detailed structures are illustrated in Figure 3 (a) (i)-(iv). The Dec-Adapter is designed to be flexible and can be inserted into every ResBlock of the decoder to transfer features from different depths. For example, it can be inserted into the residual

Algorithm 1 Compression with Per Image Adaptation

procedure LATENTREFINEMENT(y, x)
  for for step in MAX steps **do**
    $\tilde{y} := y + \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$
    $\tilde{x} := g_s(\tilde{y})$
    $L(\tilde{y}) := \sum_i -log_2 p_{\tilde{Y}_i}(\tilde{y}_i) + \lambda d(x, \tilde{x})$
    $y := y + step(L(y))$
  end for
  return: y
end procedure

procedure DECODERADAPTION($\hat{y}, \theta$)
  for for step in MAX steps **do**
    Adapters($\tilde{\theta}$) := Adapters(θ) $+ \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$
    $\tilde{x} := g_s(\tilde{y}$; Adapters($\tilde{\theta}$))
    $L(\tilde{\theta}) := \sum_i -log_2 p_{\tilde{\Theta}_i}(\tilde{\theta}_i) + \lambda d(x, \tilde{x})$
    Adapters(θ) := Adapters(θ) + $step(L(\theta))$
  end for
  return: Adapters(θ)
end procedure

procedure ENCODE(x, Adapters(θ))
  $y := g_a(x)$
  $y :=$ LatentsRefinement(y, x)
  $\hat{y} :=$ quantize(y)
  Adapters(θ) := DecoderAdaption($\hat{y}$, Adapters(θ))
  Adapters($\hat{\theta}$) := quantize(θ)
  $b := AE(\hat{y}) + AE($Adapters($\hat{\theta}$))
  return: b
end procedure

procedure DECODE(b)
  $\hat{y} := AD(b_y)$
  $\hat{x} := g_s(\hat{y}; \theta \leftarrow \theta + AD(b_\theta))$
end procedure

blocks in different stages ($s1 - s4$) of g_s, as shown in Figure 2. Other modules, such as attention blocks, can also be adapted following the same guidelines. For more details, please see our supplementary file.

3.5. Model Optimization

Our content-adaptive optimization framework is composed of two stages. The first stage is Latent Refinement while the second stage focuses on Dec-Adapter Training.

Latent Refinement. In adaptive compression, the receiver side requires access to the adapted prior and decoder models. When the amount of updated parameters is large, the cost of transmitting the model updates becomes heavy [35]. If we only adjust the encoder or latent, there are no further bits introduced to the bitstream since the encoder is not required for decoding and the latent is transmitted anyway. This allows us to close or at least reduce the amortization gap (the difference between $p_{y|z}$ and the optimal entropy p_y). Various works have explored this approach. Lu *et al.* [24] adapted the encoder, while Campos *et al.* [6], Yang *et al.* [41] directly adapted the latents. This straightforward method has been demonstrated to provide a moderate improvement in RD performance without bring any additional bit cost. Therefore, we follow Campos *et al.* [6] to refine latent representation via optimization in terms of rate-distortion at the first stage. When optimizing the latent, we freeze all other parameters and set the parameters of Dec-Adapters to zero. Specifically, given an image x, we first encode it into the latent space to obtain y. Then, we optimize the latent code y via:

$$L(y) = R(\tilde{y}) + \lambda D(g_s(\tilde{y}), x). \tag{4}$$

Here, $\tilde{y}$ represents y after uniform quantization and g_s is the decoder without adapters. Once we have completed the optimization and obtained the refined latent representation y, we can then quantize it to obtain $\hat{y}$ and encode $\hat{y}$ using entropy coding. Note that this latent refinement can be carried out using a local decoder on the sender's side.

Decoder Adaptation. The second stage is Decoder Adaptation. In this stage, we first decode the $\hat{y}$ that was compressed during the Latent Refinement stage. We then optimize the parameters of the adapters. Let w represent the quantization interval and $\hat{\theta}$ represent the quantized adapter parameters. We quantize θ with approximation and optimize θ in terms of rate-distortion. Following [34], we use a mixed quantization approach where we uniformly quantize θ with a straight-through estimator for the decoder and add uniform noise $\mathcal{U} \in (-w/2, w/2)$ to θ for the entropy model. Let $\tilde{\theta}$ represent the adapter parameters with added uniform noise. We then learn the adapter via:

$$L(\theta) = R(p_{\tilde{\theta}}) + \lambda D(g_s(\hat{y}, \hat{\theta}), x). \tag{5}$$

Table 1. The datasets used for compression evaluation.

Dataset	Domain		Test Number	Average Resolution
	Natural	Screen Content		
Kodak	✓		24	576 × 704
SIQAD		✓	24	685 × 739
SCID		✓	40	720 × 1280
CCT		✓	24	915 × 1627

The final image compression pipeline is detailed in Algorithm 1. The operations for lossless arithmetic encoding and decoding are denoted by AE and AD, respectively.

4. Experiments

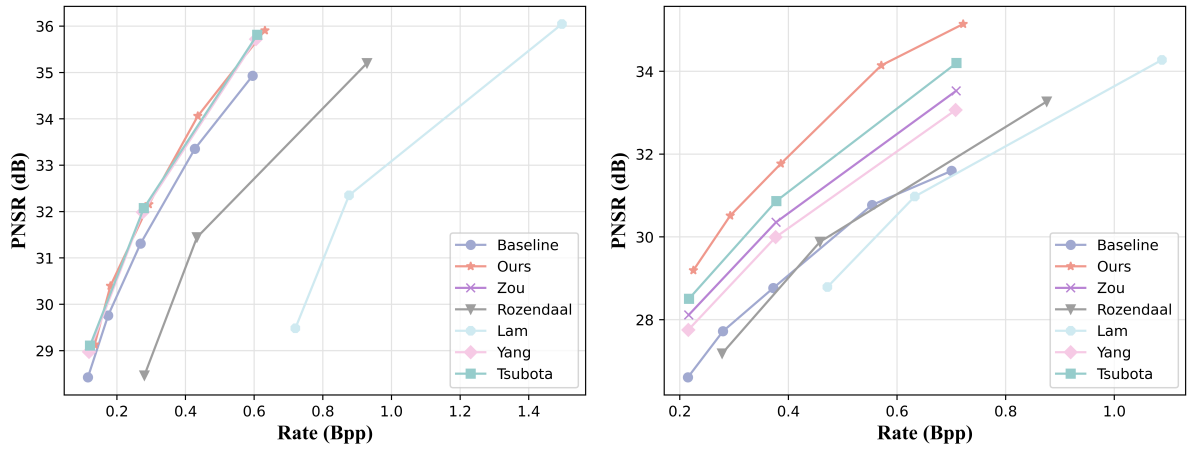
In this section, we demonstrate the superiority of Dec-Adapter's domain transfer capabilities over other relevant methods. Additionally, we provide an ablation study of Dec-Adapter's design choices and analyze its performance.

Datasets. We use Kodak [22], SCID [32], SIQAD [39], CCT [28] as our benchmark datasets. The first is natural image dataset, and the other three are screen content image datasets that consist of a combination of computer-generated graphics, text, and images. The statistics of the benchmark datasets are presented in Table 1.

We adopted Cheng2020 [8] as our backbone. The backbone was trained using natural images with various λ settings (0.0018, 0.0035, 0.0067, 0.0130, 0.0250). The pre-training data are natural images. Therefore, the in-domain performance is evaluated on the natural images and the out-of-domain performance is evaluated on the other three screen content datasets. The Adam optimizer is used for up to 2,500 iterations for the latent refinement and 500 iterations for the adapter training. For the latent refinement, we set the learning rate to 1e-3 for the first 80% of iterations and 1e-4 for the remaining 20% of iterations. The adapter training followed the same learning rate decrease strategy. We use the same value of hyper-parameters λ for both the latent refinement and decoder adaption stages as we use in the pre-training stage. We measure distortion D using mean squared error.

4.1. Comparison with State-of-the-arts

Rate-Distortion Performance. We provide four solutions for our adapter, and we use the best solution, namely Serial Convolution at location ② for the following comparison. We first compare our method with the baseline that does not use adaptive optimization. We calculate PSNR and BPP for each image and plot the average values on a rate-distortion curve, as shown in Figure 4. Our method outperforms the baseline by 2 dB when adapted to the screen content images, which verifies the effectiveness of our adaptation performance for bridging natural and screen content compression. Even on the Kodak dataset, which represents in-domain performance, our method still improves upon the



(a) Kodak (b) SCID

(c) SIQAD (d) CCT

Figure 4. Comparison with Cheng2020 [8], which is the baseline method that does not perform adaptive optimization.

Table 2. Comparison with existing adaptive compression methods in terms of BD rate (%) over Cheng2020 [8]. A smaller value means more effective.

Method	Description	Kodak (In Domain)	SIQAD (Out Domain)	CCT (Out Domain)	SCID (Out Domain)	Average
Cheng2020 [8]	Baseline	0.00	0.00	0.00	0.00	0.00
+ Yang *et al.* [41]	Only refine latent	-13.36	-25.02	-18.42	-24.38	-20.29
+ Lam *et al.* [23]	Bias-tuning	171.15	32.34	50.61	10.00	66.03
+Rozendaal *et al.* [35]	Full para. fine-tuning	63.07	5.59	16.92	-2.07	20.88
+ Zou *et al.* [46]	OMP-tuning	-13.93	-29.57	-20.72	-30.32	-23.63
+ Tsubota *et al.* [34]	MD-tuning	-14.47	-38.31	-21.05	-37.42	-27.81
+ Ours	Conv-tuning	-12.87	-47.60	-21.77	-46.27	-32.13

baseline. This confirms that our proposed adaptive scheme is effective for both natural and screen content image compression. Then, we compare our proposed method with other adaptation methods. To ensure fairness in the comparison, we integrate the methods of Yang *et al.* [41], Rozendaal *et al.* [35], Zou *et al.* [46], Lam *et al.* [23] and Tsubota *et al.* [34] into the decoder of our baseline. For more information on the experimental setup, please refer to the supplementary file. We calculate the Bjøntegaard Delta bitrate (BD rate) [5] in relation to our baseline Cheng2020 [8]. This metric is used to evaluate the performance of video (image) encoders and is primarily employed to compare the performance of two encoders at the same bit rate. A lower BD rate indicates better performance. The results are listed in Table 2. It is clear that our proposed method outperforms other adaptation methods, particularly when applied to out-

of-domain contents. When compressing Kodak images (in-domain), our method performs slightly below the state-of-the-art method Tsubota *et al.* [34]. The main reason is that our adaptation needs to encode the parameter of the adapter and this does not bring gains for in-domain image adaptation. When adapt to the screen content compression, our Conv-based Tuning adapters clearly outperforms the MD (Matrix Decompose)-tuning method of Tsubota *et al.* and the OMP (overfittable multiplicative parameters) method of Zou *et al.* [46]. It should be noted that Full parameter-based tuning [35] and Bias-based tuning perform [23] worse than the baseline method. This is because these methods transmit a large number of parameters to the decoder for each individual image. This also demonstrates that the Conv based adapter, which captures information effectively with a small number of parameters, is the best choice. Table 3 further lists the results of all the tuning methods based on a stronger baseline WACNN [47]. We insert our adapter into the last layer of the decoder following the setting of [34]. The DW-conv kernel K is set to 3 and the bottleneck layer dimension is 16, which is the same as for ResBlock architectures. Compared with this baseline, our method still achieves the best BD-rate with -13.52% (if averaged on the three SC testsets, the BD-rate is -19.75%).

Table 3. Comparison with adaptive compression methods with WACNN [47] as the network backbone.

	Kodak	SIQAD	CCT	SCID	Average
Baseline(WACNN)	0.00	0.00	0.00	0.00	0.00
+ Yang *et al.* [41]	-0.74	-8.10	-5.74	-10.44	-6.26
+ Lam *et al.* [23]	185.8	69.52	66.00	47.68	92.25
+ Rozendaal *et al.* [35]	74.81	41.51	33.23	35.86	46.34
+ Zou *et al.* [46]	-4.74	-9.38	-8.37	-7.62	-7.53
+ Tsubota *et al.* [34]	-5.32	-15.22	-8.78	-10.56	-9.97
+ Ours	5.18	-23.17	-11.44	-24.65	-13.52

Qualitative Results. Figure 5 illustrates qualitative results from our proposed method alongside the baseline and five adaptation methods at comparable BPP levels. All the results are generated with Cheng2020 [8] as the baseline. It can be observed that our method can recover sharp edges well and the characters are also well recovered.

4.2. Ablation Study

We perform ablation study on the design choices of Dec-Adapter. We explore different architectures and adapting schemes to evaluate their effectiveness. Our results are reported as the BD rate (%) over the baseline on the SIQAD dataset.

Adapting variants exploration. We first compare the performances of four adapting variants. Tables 4 and 5 present the results of four different connection methods when connecting at ① and ②, as shown in Fig. 3: Par-Conv-1/2, Serial-Conv-1/2, Serial-Conv w/o id-1/2 and Par-Res. We report the results in terms of PSNR (Peak Signal-to-Noise Ratio) and BPP (Bits Per Pixel) for three different quality levels as well as the BD rate. The three quality level (q1, q3 and q5) corresponds to setting λ to (0.0018, 0.0067, 0.0250). At the three quality levels for both cases, the Serial-Conv w/o identity method performs much worse than the other three methods in terms of PSNR. This suggest that residual information plays an important role in transfer learning for cross-domain decompression. Among the other three methods for both ① and ②, Serial-Conv has the lowest BD-rate, indicating that it performs better than both Par-Conv and Par-Res. Additionally, Serial-Conv performs better at location ② than at location ①. Therefore, we choose location ② as the default connection point for our adapter.

Table 4. Comparison (PSNR/BPP) of four connection methods when connecting at ①.

	Par-Conv-①	Serial-Conv-①	Serial-Conv w/o id-①	Par-Res
q1	27.10/0.3003	27.13/0.2998	18.06/0.2987	27.15/0.3002
q3	29.54/0.5130	29.68/0.5125	18.62/0.5113	29.64/0.5127
q5	32.70/0.9391	32.62/0.9384	19.18/0.9367	32.62/0.9385
BD-rate(%)	-35.14	-36.71	92910	-36.41

Table 5. Comparison (PSNR/BPP) of four connection methods when connecting at ②.

	Par-Conv-②	Serial-Conv-②	Serial-Conv- w/o id-②	Par-Res
q1	27.15/0.3004	27.28/0.3002	18.15/0.2998	27.15/0.3002
q3	29.67/0.5131	29.84/0.5127	18.70/0.5129	29.64/0.5127
q5	32.68/0.9395	32.86/0.9386	19.53/0.9418	32.62/0.9385
BD-rate(%)	-36.66	-38.83	241.6	-36.41

Sensitivity to DWConv kernel K and channel squeeze factor α. We study the sensitivity of transfer performance to the DWConv kernel K and squeeze factor α in Dec-Adapter (Figure 6). Dec-Adapter achieves the best BD-rate performance on the SIQAD dataset when $K = 3$. Increasing the kernel size degrades BD-rate because the additional parameters increase larger BPP but bring smaller PSNR gains. Decreasing squeeze factor α, or increasing bottleneck layer channels, improves the BD-rate performance until bottleneck layer dimension reaches 16. Setting bottleneck layer channels C/α with a larger value results in inferior performance. Therefore, unless otherwise specified, we choose DWConv kernel $K = 3$ and bottleneck layer channels $C/\alpha = 16$ as Dec-Adapter's default choice.

Layer analysis for adapters. We investigate whether attaching adapters only to later layers is sufficient. We evaluate this on the decoder of Cheng2020 [8], which has four blocks. We attach adapters to each block (block1, block2, block3, block4) and combinations of blocks (block1-2-3-4), see Fig. 2. Fig. 7 shows that applying adapters only to the last block (block4) achieves over 20% BD-rate savings (averaged on all domains). Attaching adapters to the last layer significantly improves performance on unseen domains while maintaining stable performance on seen domains. Adding adapters to all four blocks can further improve the performance, but the gain is limited. Therefore, to save the computation cost, we choose to add adapters only to the last layer.

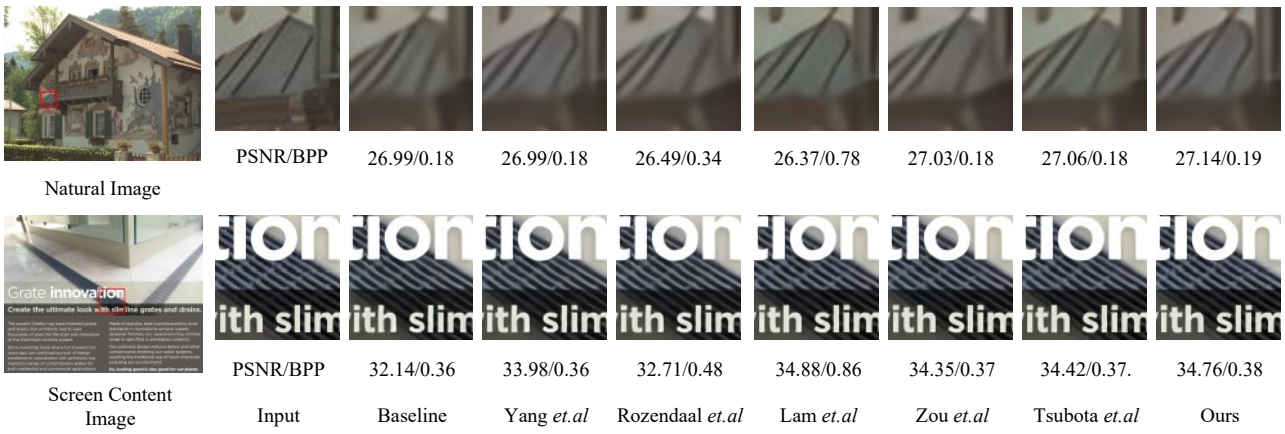


Figure 5. Qualitative results for different domains.

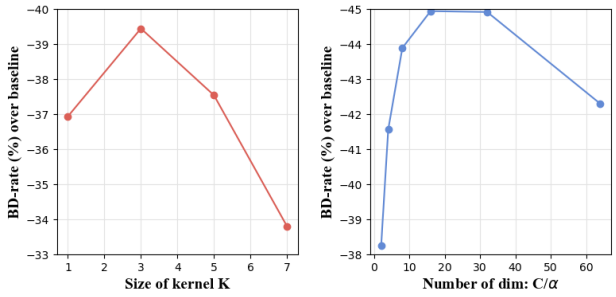


Figure 6. Sensitivity to DWConv Kernel K and compression factor α.

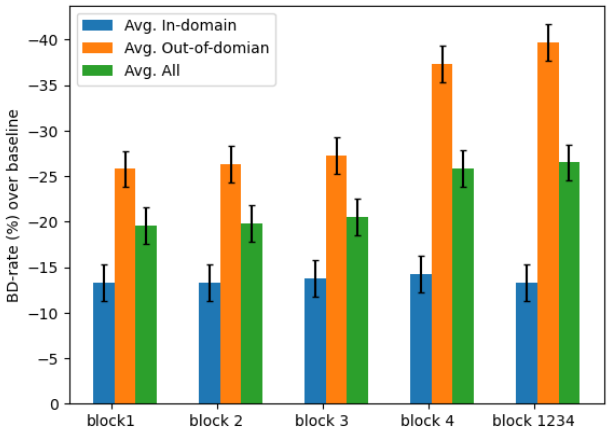


Figure 7. Block (layer) analysis for adapters.

4.3. Comparison with HEVC-SCC and VVC-SCC

Further, we also compare with HEVC-SCC Intra (HM 18.0) [1], and VVC Intra (VTM 20.0) [2] with screen content coding options enabled. Table 6 presents the comparison results. Since our method considers both SC and natural images, our method still has a large gap with the traditional SCC standards. However, as we incorporate a more powerful backbone, this gap is steadily narrowing. We believe our work can inspire more works on SCC.

Table 6. Comparison with HEVC-SCC and VVC-SCC.

	Kodak	SIQAD	CCT	SCID	Average
Cheng2020	0.00	0.00	0.00	0.00	0.00
Cheng2020 + Ours	-12.87	-47.60	-21.77	-46.27	-32.13
WACNN + Ours	-6.70	-54.02	-22.62	-57.00	-35.09
HEVC-SCC	25.90	-67.91	-53.59	-74.98	-42.64
VVC-SCC	-25.37	-73.35	-27.08	-75.04	-50.20

5. Conclusion

In conclusion, this paper proposes a new method, namely Dec-Adapter, for bridging natural and screen content image compression. It is a entropy-efficient transfer learning module designed for the decoder. The adapter's parameters are learned during encoding and transmitted to the decoder for image-adaptive decoding. We provide a comprehensive analysis on the design of the adapter structure, the decoder structure, and the insertion position. Our method outperforms existing methods in terms of BD-rate performance on screen content image compression and achieves an improvement of over 2 dB to the baseline.

6. Acknowledgments

The authors deeply appreciate the constructive suggestions provided by the reviewers.

References

[1] M. Abdoli, G. Clare, and F. Henry. Gop-based latent refinement for learned video coding. In *ICASSP 2023 - 2023 IEEE*

[1] https://vcgit.hhi.fraunhofer.de/jvet/HM

[2] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM

International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5, 2023. 2

[2] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *International Conference on Learning Representations*, 2016. 1, 2

[3] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2

[4] F. Bellard. Bpg image format. *URL https://bellard. org/bpg*, 1(2):1, 2015. 1

[5] G. Bjontegaard. Calculation of average psnr differences between rd-curves. *ITU SG16 Doc. VCEG-M33*, 2001. 6

[6] D. S. Campos, Simon. Content adaptive optimization for neural image compression. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, 2019. 5

[7] H. Chen, R. Tao, H. Zhang, Y. Wang, W. Ye, J. Wang, G. Hu, and M. Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets. *arXiv preprint arXiv:2208.07463*, 2022. 2, 3

[8] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948, 2020. 1, 2, 3, 5, 6, 7

[9] Y. Choi, M. El-Khamy, and J. Lee. Variable rate deep image compression with a conditional autoencoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3146–3154, 2019. 2

[10] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018. 2

[11] H. de Jesús Ochoa Domínguez and K. R. Rao. Versatile video coding. 2019. 1

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[13] J. Djelouah and C. Schroers. Content adaptive optimization for neural image compression. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, 2019. 2

[14] C. Gao, T. Xu, D. He, Y. Wang, and H. Qin. Flexible neural image compression via code editing. *Advances in Neural Information Processing Systems*, 35:12184–12196, 2022. 2

[15] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5718–5727, 2022. 2

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[17] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. H. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. *International Conference on Machine Learning*, 2019. 2, 3

[18] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 3

[19] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim. Visual prompt tuning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 709–727. Springer, 2022. 3

[20] O. Jourairi, M. Balcilar, A. Lambert, and F. Schnitzler. Improving the reconstruction quality by overfitted decoder bias in neural image compression. In *2022 Picture Coding Symposium (PCS)*, pages 61–65, 2022. 2

[21] Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687. PMLR, 2018. 2

[22] E. Kodak. Kodak lossless true color image suite (photocd pcd0992). *URL http://r0k. us/graphics/kodak*, 6, 1993. 2, 5

[23] Y.-H. Lam, A. Zare, F. Cricri, J. Lainema, and M. M. Hannuksela. Efficient adaptation of neural network filter for video compression. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 358–366, 2020. 1, 2, 6, 7

[24] G. Lu, C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, and Z. Gao. Content adaptive and error propagation aware deep video compression. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 456–472. Springer, 2020. 2, 5

[25] G. Luo, M. Huang, Y. Zhou, X. Sun, G. Jiang, Z. Wang, and R. Ji. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106*, 2023. 2, 3

[26] J. Marino, Y. Yue, and S. Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018. 2

[27] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems*, 33:11913–11924, 2020. 2

[28] X. Min, K. Ma, K. Gu, G. Zhai, Z. Wang, and W. Lin. Unified blind quality assessment of compressed natural, graphic, and screen content images. *IEEE Transactions on Image Processing*, 26(11):5462–5474, 2017. 5

[29] D. Minnen, J. Ballé, and G. Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10794–10803, 2018. 1

[30] D. Minnen and S. Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3339–3343. IEEE, 2020. 2

[31] D. Mishra, S. K. Singh, and R. K. Singh. Deep architectures for image compression: a critical review. *Signal Processing*, 191:108346, 2022. 2

[32] Z. Ni, L. Ma, H. Zeng, J. Chen, C. Cai, and K.-K. Ma. Esim: Edge similarity for screen content image quality assessment.

IEEE Transactions on Image Processing, 26(10):4818–4831, 2017. 5

[33] G. Pan, G. Lu, Z. Hu, and D. Xu. Content adaptive latents and decoder for neural image compression. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVIII*, pages 556–573. Springer, 2022. 2

[34] K. Tsubota, H. Akutsu, and K. Aizawa. Universal deep image compression via content-adaptive optimization with adapters. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2529–2538, 2023. 1, 2, 5, 6, 7

[35] T. van Rozendaal, I. A. Huijben, and T. S. Cohen. Overfitting for fun and profit: Instance-adaptive data compression. *arXiv preprint arXiv:2101.08687*, 2021. 1, 2, 5, 6, 7

[36] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 1992. 1

[37] M. Wang, K. Zhang, L. Zhang, Y. Wu, Y. Li, J. Li, and S. Wang. Transform skip inspired end-to-end compression for screen content image. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3848–3852. IEEE, 2022. 2

[38] T. Xu, H. Gao, C. Gao, Y. Wang, D. He, J. Pi, J. Luo, Z. Zhu, M. Ye, H. Qin, et al. Bit allocation using optimization. In *International Conference on Machine Learning*, pages 38377–38399. PMLR, 2023. 2

[39] H. Yang, Y. Fang, and W. Lin. Perceptual quality assessment of screen content images. *IEEE Transactions on Image Processing*, 24(11):4408–4421, 2015. 1, 5

[40] T. Yang, Y. Zhu, Y. Xie, A. Zhang, C. Chen, and M. Li. Aim: Adapting image models for efficient video action recognition. *arXiv preprint arXiv:2302.03024*, 2023. 2, 3

[41] Y. Yang, R. Bamler, and S. Mandt. Improving inference for neural image compression. *Advances in Neural Information Processing Systems*, 33:573–584, 2020. 2, 5, 6, 7

[42] Y. Yang and S. Mandt. Asymmetrically-powered neural image compression with shallow decoders. *arXiv preprint arXiv:2304.06244*, 2023. 2

[43] R. Zamanshoar Heris and I. V. Bajić. Multi-task learning for screen content image coding. *arXiv e-prints*, pages arXiv–2302, 2023. 2

[44] P. Zhang, S. Wang, M. Wang, J. Li, X. Wang, and S. Kwong. Rethinking semantic image compression: Scalable representation with cross-modality transfer. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023. 2

[45] J. Zhao, B. Li, J. Li, R. Xiong, and Y. Lu. A universal encoder rate distortion optimization framework for learned compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1880–1884, 2021. 2

[46] N. Zou, H. Zhang, F. Cricri, R. G. Youvalari, H. R. Tavakoli, J. Lainema, E. Aksu, M. Hannuksela, and E. Rahtu. Adaptation and attention for neural video coding. In *2021 IEEE International Symposium on Multimedia (ISM)*, pages 240–244. IEEE, 2021. 1, 2, 6, 7

[47] R. Zou, C. Song, and Z. Zhang. The devil is in the details: Window-based attention for image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17492–17501, 2022. 1, 2, 7