Universal Deep Image Compression via Content-Adaptive Optimization with Adapters

Koki Tsubota[1] Hiroaki Akutsu[2] Kiyoharu Aizawa[1]
[1]The University of Tokyo [2]Hitachi, Ltd.
{tsubota,aizawa}@hal.t.u-tokyo.ac.jp, hiroaki.akutsu.cs.@hitachi.com

Abstract

Deep image compression performs better than conventional codecs, such as JPEG, on natural images. However, deep image compression is learning-based and encounters a problem: the compression performance deteriorates significantly for out-of-domain images. In this study, we highlight this problem and address a novel task: universal deep image compression. This task aims to compress images belonging to arbitrary domains, such as natural images, line drawings, and comics. To address this problem, we propose a content-adaptive optimization framework; this framework uses a pre-trained compression model and adapts the model to a target image during compression. Adapters are inserted into the decoder of the model. For each input image, our framework optimizes the latent representation extracted by the encoder and the adapter parameters in terms of rate-distortion. The adapter parameters are additionally transmitted per image. For the experiments, a benchmark dataset containing uncompressed images of four domains (natural images, line drawings, comics, and vector arts) is constructed and the proposed universal deep compression is evaluated. Finally, the proposed model is compared with non-adaptive and existing adaptive compression models. The comparison reveals that the proposed model outperforms these. The code and dataset are publicly available at `https://github.com/kktsubota/universal-dic`.

1. Introduction

Image compression is a fundamental technology for reducing the costs of storage and network transmission. Compressed images are ubiquitous—digital cameras and smartphones compress images. The common compression standard is JPEG [51], whereas JPEG2000 [45], BPG [8], and VVC [10] are more recent standard-based compression. Deep image compression is the image compression technique based on neural networks. Recent studies have demonstrated that deep image compression exhibits higher performance than conventional codecs on natural images [19, 54, 57].

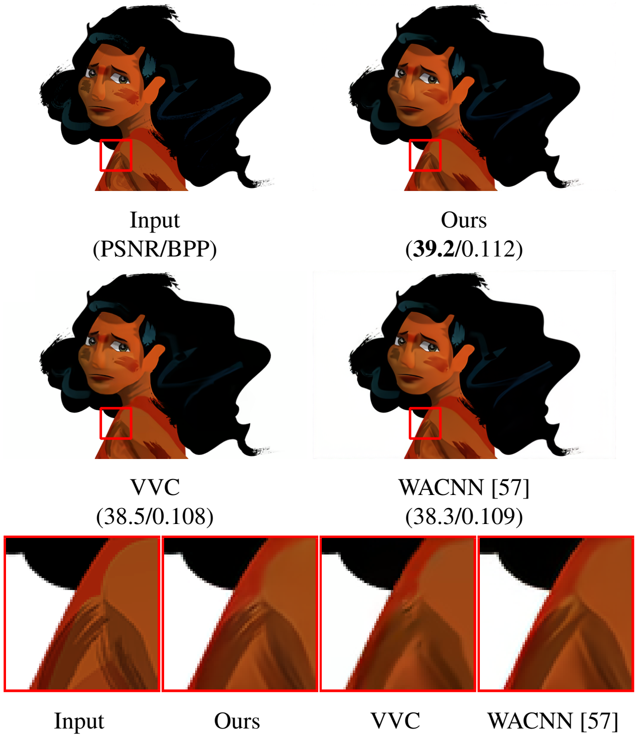


Figure 1: Examples of compression results on a comic image in the BAM dataset [52]. General deep image compression (WACNN [57]) performs superior to the state-of-the-art conventional codec (VVC [10]) on natural images. However, its performance deteriorates on out-of-domain images. By addressing this problem, our framework exhibits superior performance to VVC on out-of-domain images. In this figure, ours can reconstruct the brush texture in dark brown with relatively high fidelity.

However, deep image compression is learning-based. Therefore, we encounter the problem of performance degradation in compressing out-of-domain images. General compression models pre-trained only on natural images exhibit relatively low performance on images in other domains, as shown in Fig. 1. To investigate this problem, we address a novel deep image compression task, which we name universal deep image compression. The objective of universal deep image compression is to compress images from arbitrary domains, such as line drawings and comics, as well as natural images.

We propose a content-adaptive optimization framework to address the problem of compressing out-of-domain images. This framework adapts the pre-trained compression model to each target image and addresses domain shifts between pre-training and testing. Our framework is efficient owing to a small number of parameters needed for the per-image adaptation during testing.

Our framework has two advantages over previous approaches studied in content-adaptive compression [13, 30, 31, 48, 55, 56]: the flexibility of the base network architecture and the efficiency in terms of rate-distortion. In content-adaptive compression, certain studies adapted the compression model during testing by refining the latent representation extracted by the encoder [13, 53]. Other studies additionally updated the parameters in the decoder and transmitted these [30, 31, 48, 55, 56]. However, the state-of-the-art latent refinement method [53] has restrictions on pre-trained compression models: it assumes that the hyper latent representation follows a Gaussian distribution to perform the bit-back coding [22, 50]. Moreover, previous approaches for updating the parameters in the decoder update an excessive number of parameters for an individual image [31, 48], insert and train ad-hoc layers [56], or optimize parameters only in terms of distortion [31, 56].

In our framework, we refine the latent representation by the simplified approach of the state-of-the-art refinement method. We omit the process of bit-back coding and only optimize the latent representation in terms of rate-distortion with gradient descent. Therefore, our latent refinement is efficient and flexible to the base network architecture.

To update the decoder, we insert adapters into the decoder and train these. Adapters are small modules with a small number of parameters and have been successful in parameter-efficient transfer learning [23, 33, 41, 46]. Using adapters, we can improve the compression performance by updating a relatively small number of parameters. Moreover, we optimize adapters in terms of rate-distortion with gradient descent. Therefore, our decoder update is efficient in terms of rate-distortion.

To evaluate our framework, we construct a benchmark dataset that comprises four domains: natural images, line drawings, comics, and vector arts. We sample natural images from the Kodak dataset [16] and images in the other three domains from the BAM dataset [52]. We use one of the state-of-the-art compression models (window attention-based convolutional neural networks (WACNN) [57]) for the baseline, and modify it by inserting adapters and optimizing the latent representation and the adapter parameters. We pre-train the model on a natural image dataset and evaluate its performance on in-domain and out-of-domain images.

The main contributions of this study are as follows

- We address universal deep image compression. To our knowledge, this is the first work that addresses the deep compression of images in arbitrary domains, such as line drawings and comics.
- We propose a content-adaptive optimization framework, wherein we adapt a pre-trained model to each target image. Our framework refines the latent representation by a simplified approach of the state-of-the-art method. We then train adapters inserted into the decoder via optimization in terms of rate-distortion. The adapter parameters are additionally transmitted.
- We demonstrate experimentally that the proposed method is effective and significantly outperforms the state-of-the-art conventional codec on the four domains.

2. Related Work

2.1. Deep Image Compression

Deep image compression achieves image compression by optimizing the modules in an end-to-end manner [5]. To obtain compressed images with less distortion, numerous studies have worked toward improving the modules such as the encoders and decoders [14, 54] and the entropy models [34, 36, 37]. Some studies worked on image compression for human perception instead of distortion [12, 35, 39, 43]. Other studies worked on achieving variable rate compression [15].

Deep image compression outperforms conventional codecs on natural images [19, 54, 57]. However, these studies trained only on natural images, such as CLIC [47] and ImageNet [44], and evaluated only on natural images, such as Kodak [16], CLIC [47], Tecnick [3], and DIV2K [2]. Hence, their compression performance when applied to other domains, such as line drawings and comics, remains uncertain.

In contrast, Kim *et al*. [27] worked on lightweight and fast decoding in deep image compression. They evaluated their proposed method on both natural and cartoon images. However, their approach needs to prepare a dataset for training on cartoon images, unlike a content-adaptive optimization framework that requires no pre-training per domain. Moreover, the number of evaluation domains is limited. In this study, we study the compression of images in various domains, such as line drawings, comics, and natural images.

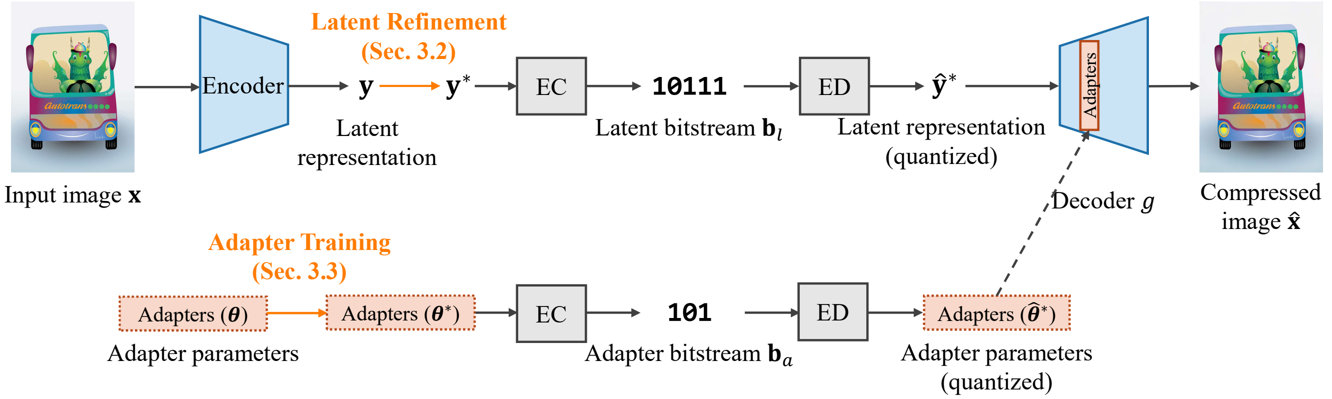


Figure 2: Outline of the proposed method. First, we refine the latent representation; subsequently, we train the adapters.

2.2. Content-Adaptive Compression

Content-adaptive compression achieves high performance by adapting the compression model for each target image. In deep image compression, content-adaptive compression is achieved by the per-image refinement of the latent representation obtained by the encoder [13, 53]. The latent bitstream can be obtained by compressing the refined latent representation. In addition to the latent refinement, Zou *et al*. [55] updated the parameters in the decoder per image. The model bitstream can be obtained by compressing the updated parameters.

Updating and compressing the parameters in the decoder has been studied primarily for compression of multiple images, post-processing of video compression, and deep video compression. Rozendaal *et al*. [48] updated all the parameters in the decoder and entropy model. They optimized the parameters in terms of rate-distortion and finally compressed these by entropy coding. However, although they addressed multiple images for the adaptation, this approach requires a relatively large number of bits for compressing an individual image.

Other studies updated the limited number of adapting parameters. Zou *et al*. [55] addressed deep image compression and updated only the biases of convolution layers in the decoder. Lam *et al*. [31] addressed the post-processing of compressed videos and updated only the biases of convolutional layers in the post-processing network. Zou *et al*. [56] addressed deep video compression. They inserted overfittable multiplicative parameters (which multiply the output of convolutional layers) and updated these for intra-frame coding. However, updating parameters are selected in an ad-hoc manner and are optimized only in terms of distortion. Unlike these previous approaches, we introduce adapters and optimize these in terms of rate-distortion. Adapters exhibit superior performance in parameter-efficient transfer learning.

2.3. Parameter-Efficient Transfer Learning

Parameter-efficient transfer learning aims to adapt a model pre-trained on a large-scale dataset per task, with reducing the number of adapting parameters. Unlike a general adaptation approach that fine-tunes all the parameters of a pre-trained model, most of the model parameters are fixed. Therefore, we can reduce the cost of transmission of parameters and preserve the knowledge in pre-training.

This task was first studied for obtaining a universal representation of multiple domains in computer vision [41, 42]. Recently, motivated by the emergence of a big pre-trained transformer model [49], such as BERT [17] and T5 [40], this task has been studied primarily for training efficiently in natural image processing [9, 23].

We can classify the algorithms for parameter-efficient transfer learning into the following two types. (1) adapting newly added parameters [20, 25, 33, 41, 42] and (2) adapting part of the parameters in the model [9, 18, 38]. The first approach introduces an adaptation module, such as adapters [41, 42], compacters [25], and hyperformers [26]. The second approach adapts normalization and lightweight layers [38], biases of layers in the model [9], and sparse difference in model weights [18].

Among these approaches, adapters are widely used [20, 23, 33, 41, 42, 46]. Adapters are modules with a small number of parameters. These are implemented as matrix multiplication [42], decomposition of a matrix [42], multiplication of two matrices with activation [23], or channel-wise scaling [33]. Motivated by the success of adapters in this task, we used adapters in our framework.

3. Method

The objective of universal deep image compression is to compress images from arbitrary domains. To achieve this objective, we proposed a content-adaptive optimization framework. Given a pre-trained compression model and an uncompressed image, the model is adapted to each target

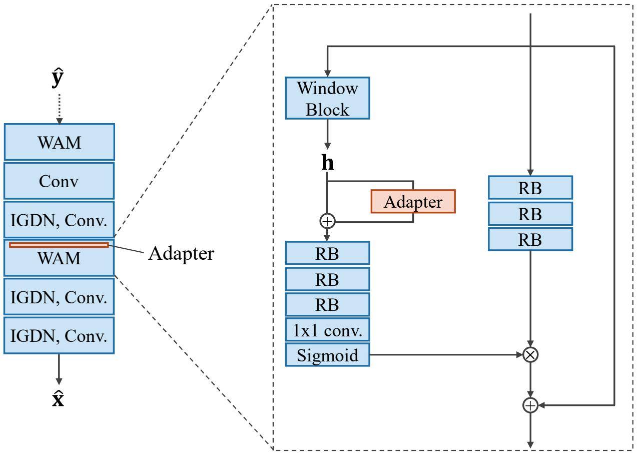


Figure 3: Network architecture of the main decoder with adapters. WAM, Conv., IGDN, and RB denote a window-attention module [57], a convolutional layer, an inverse generalizable divisible normalization layer [4], and a residual block [21], respectively.

image during testing.

The outline of the proposed method is shown in Fig. 2. In encoding, first, latent representation is extracted by the encoder and adapters are inserted into the decoder. Subsequently, the latent representation is refined via optimization in terms of rate-distortion. Next, the adapters are trained by optimizing in terms of rate-distortion. Finally, the latent representation and the parameters of the adapters are encoded by entropy coding, and the latent and adapter bitstreams are transmitted. In decoding, the transmitted bitstreams are decoded by entropy decoding, and finally, the compressed image is obtained.

We used WACNN [57], which is a state-of-the-art architecture, as the base network architecture. Note that our framework can be applied to other network architectures. WACNN has a hyper-prior architecture [6]. It transmits the hyper latent representation as well as the latent representation. In our explanation, we considered these two representations in a unified manner and called these as the latent representation.

Hereafter, we explain the details of our framework. Our framework comprises the following three technical components. The first is the insertion of the adapters into the decoder. The second is the refinement of the latent representation extracted from the target image. The third is the training of the adapters. We describe the details of each component in the next subsections.

Let us define the characters for the explanation. Let $\mathbf{x}$ be the input image, $\hat{\mathbf{x}}$ be the compressed image, $\mathbf{y}$ be the latent representation, $\hat{\mathbf{y}}$ be the quantized latent representation, $\mathbf{y}^*$ be the refined latent representation, q be the quantizer, g be the decoder with adapters, $\mathbf{b}_l$ be the latent bitstream, $\mathbf{b}_a$ be the adapter bitstream, $\boldsymbol{\phi}$ be the pre-trained parameters for the decoder except for adapters, $\boldsymbol{\theta}$ be the parameters of the adapters, and $\boldsymbol{\theta}^*$ be the updated parameters of the adapters.

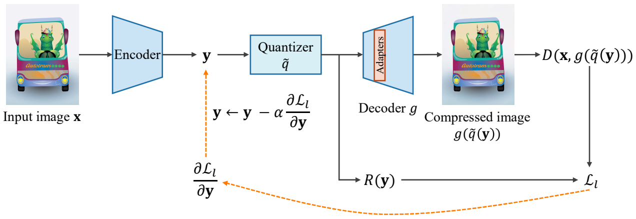

Input image $\mathbf{x}$
Encoder
$\mathbf{y}$
Quantizer $\tilde{q}$
Adapters
Decoder g
Compressed image $g(\tilde{q}(\mathbf{y}))$
$D(\mathbf{x}, g(\tilde{q}(\mathbf{y})))$
$\mathbf{y} \leftarrow \mathbf{y} - \alpha \frac{\partial \mathcal{L}_l}{\partial \mathbf{y}}$
$\frac{\partial \mathcal{L}_l}{\partial \mathbf{y}}$
$R(\mathbf{y})$
$\mathcal{L}_l$

Figure 4: Outline of latent refinement.

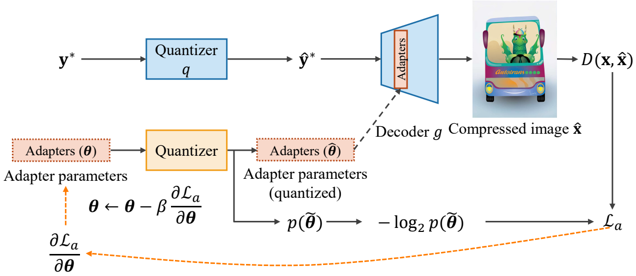

$\mathbf{y}^*$
Quantizer q
$\hat{\mathbf{y}}^*$
Adapters
Decoder g
Compressed image $\hat{\mathbf{x}}$
$D(\mathbf{x}, \hat{\mathbf{x}})$
Adapters ($\boldsymbol{\theta}$)
Adapter parameters
Quantizer
Adapters ($\tilde{\boldsymbol{\theta}}$)
Adapter parameters (quantized)
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \frac{\partial \mathcal{L}_a}{\partial \boldsymbol{\theta}}$
$p(\tilde{\boldsymbol{\theta}})$
$-\log_2 p(\tilde{\boldsymbol{\theta}})$
$\mathcal{L}_a$
$\frac{\partial \mathcal{L}_a}{\partial \boldsymbol{\theta}}$

Figure 5: Outline of adapter training.

3.1. Insertion of Adapters

We empirically determined good insertion positions of the adapters and inserted an adapter into the window attention module (WAM) [57] on the second side of the main decoder. The network architecture of the main decoder with adapters is shown in Fig. 3.

We implemented the adapter by matrix decomposition presented in [42]. This architecture is simple albeit effective, as shown in [33]. Let the input of the adapter be $\mathbf{h} \in \mathbb{R}^{C \times H \times W}$ and the adapter be $r : \mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^{C \times H \times W}$. The operation of the adapter is written as

$$r(\mathbf{h}; \boldsymbol{\theta}) = \mathbf{A}\mathbf{B}^\top \mathbf{h}, \tag{1}$$

where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{C \times M}$ are learnable parameters of the adapter and $\boldsymbol{\theta} = [\mathbf{A}, \mathbf{B}]$. Thus, the number of adapter parameters is $2MC$.

The number of adapter parameters is much smaller than the number of model parameters. For WACNN [57], the number of model parameters is 6.50×10^7 and $C = 192$. Thus, when $M = 2$, the number of adapter parameters is 768. This is 0.0012% to the model parameters.

3.2. Latent Refinement

The outline of the latent refinement is illustrated in Fig. 4. We optimized the latent representation $\mathbf{y}$ with gradient descent via the simplified approach of Yang *et al.* [53]. The loss function is written as

$$\mathcal{L}_l(\mathbf{y}) = R(\tilde{q}(\mathbf{y})) + \lambda D\left(g\left(\tilde{q}(\mathbf{y}); \boldsymbol{\phi}, \boldsymbol{\theta}\right), \mathbf{x}\right), \tag{2}$$

where R is bitrate, D is the distortion, $\lambda \in \mathbb{R}$ is the hyperparameter for adjusting the trade-off of rate-distortion, and $\tilde{q}$ is the uniform quantization with the approximation of stochastic Gumbel annneling [53]. In this optimization, we did not train the adapters and fix the parameters to zero. We omitted bit-back coding in Yang *et al.* [53] because it required modification of the pre-trained network architecture.

We obtained the refined latent representation $\mathbf{y}^*$, which is given as follows.

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \mathcal{L}_l(\mathbf{y}). \tag{3}$$

Finally, we quantized $\mathbf{y}^*$ and encoded $\hat{\mathbf{y}}^*$ by entropy coding to obtain $\mathbf{b}_l$. Note that this latent refinement can be completed using a local decoder at the transmitter.

3.3. Adapter Training

The outline of adapter training is shown in Fig. 5. We optimized the parameters of the adapters as the optimization of the latent representation. Let w be the quantization interval and $\hat{\boldsymbol{\theta}}$ be the quantized adapter parameters. We quantized $\boldsymbol{\theta}$ with approximation and optimized $\boldsymbol{\theta}$ in terms of rate-distortion. We used the mixed quantization approach [32]. That is, we uniformly quantized $\boldsymbol{\theta}$ with a straight-through estimator [24] for the decoder and add uniform noise $U(-w/2, w/2)$ to $\boldsymbol{\theta}$ for the entropy model. Let $\tilde{\boldsymbol{\theta}}$ be the adapter parameters added uniform noise.

The loss function for optimizing $\boldsymbol{\theta}$ is written as

$$\mathcal{L}_a(\boldsymbol{\theta}) = -\log_2 p(\tilde{\boldsymbol{\theta}}) + \lambda D\left(g\left(\hat{\mathbf{y}}; \boldsymbol{\phi}, \hat{\boldsymbol{\theta}}\right), \mathbf{x}\right), \tag{4}$$

where p is the entropy model of $\boldsymbol{\theta}$. The first term is the loss function for the bitrate and computes the entropy of $\hat{\boldsymbol{\theta}}$. We used a logistic distribution with the scale of $s \in \mathbb{R}$ as p.

After this optimization, we obtained the updated parameters of the adapters $\boldsymbol{\theta}^*$, which are given as follows.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_a(\boldsymbol{\theta}). \tag{5}$$

Finally, we quantized $\boldsymbol{\theta}^*$ and encoded $\hat{\boldsymbol{\theta}}^*$ by entropy coding to obtain $\mathbf{b}_a$.

4. Experiment

4.1. Experimental Setup

We constructed a benchmark dataset containing uncompressed images of four domains: natural images, comics, line drawings, and vector arts. We collected natural images from the Kodak dataset [16] and the images in the other domains from the BAM dataset [52]. The Kodak dataset consists of 24 natural images. We used all the images in the Kodak dataset. The BAM dataset consists of artistic images other than natural images. We sampled 100 images from the BAM dataset per domain that were not degraded by JPEG compression. We considered images labeled with “pen-ink”, “comic”, and “vectorart” as line drawings, comics, and vector arts, respectively. The statistics of the constructed dataset are presented in Table 1.

Table 1: Evaluation dataset.

Domain	Test data	Average Resolution
Natural image	24	576 × 704
Comic	100	606 × 587
Line drawing	100	584 × 577
Vector art	100	554 × 580

We used WACNN [57] implemented with CompressAI [7] as the base compression model. We pre-trained six models with $\mathcal{L} = R + \lambda D$ by setting λ to 0.0018, 0.0035, 0.0067, 0.013, 0.025, and 0.0483. The pre-training data comprised 300,000 natural images sampled randomly from OpenImages [29]. Thus, the results for the natural images indicate in-domain performance, whereas those for the other three domains indicate out-of-domain performance.

With regard to the hyper-parameters, we set $w = 0.06$, $s = 0.05$, and $M = 2$ in the proposed method. We set λ to an equal value in pre-training. We use mean squared error as distortion D.

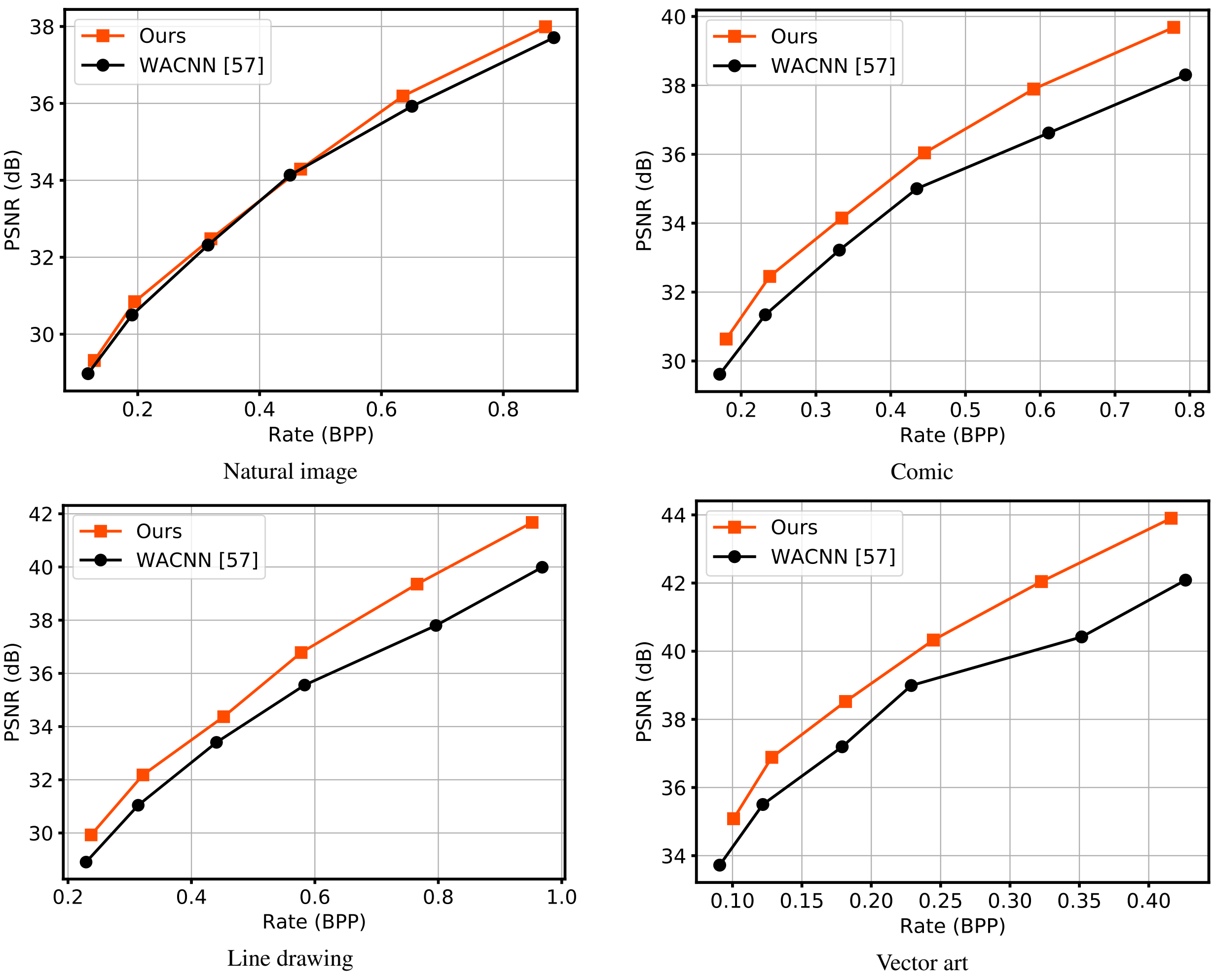
Implementation Details. In pre-training, we used the Adam optimizer [28] for up to 100 epochs with a batch size of 16. We set the learning rate to 10^{-3} for the first 78 epochs, 10^{-4} for the following 20 epochs, and 10^{-5} for the final two epochs.

In adaptation, we used the Adam optimizer for up to 2,000 iterations for the latent refinement and 500 iterations for the adapter training. We set the learning rate to 10^{-3} for the first 1,600 iterations and 10^{-4} for the final 400 iterations for the latent refinement. We set the learning rate to 10^{-3} for the first 400 iterations and 10^{-4} for the final 100 iterations for the adapter training. The $\boldsymbol{\theta}_a$ is initialized with the Gaussian noise of $\mathcal{N}(0, 0.02^2)$. For further details, please refer to our publicly available source code.

4.2. Comparison with Other Methods

Rate-Distortion Performance. First, we compared the proposed method with a baseline method that does not perform adaptive optimization. We calculated the peak signal-to-noise ratio (PSNR) and bits per pixel (BPP) for each image, and computed the average values to plot on a rate-distortion curve. The results are displayed in Fig. 6. Evidently, the proposed method significantly outperformed the baseline method. The improvement in PSNR was approximately 1–2 dB. This indicates that adaptive optimization is effective for universal deep image compression.

Next, we compared the proposed method with other



Natural image

Comic

Line drawing

Vector art

Figure 6: Comparison with WACNN [57], which is the baseline method that does not perform adaptive optimization.

Table 2: Comparison with existing adaptive compression methods on BD rate (%) to VVC [10]. The BD rates of JPEG, BPG, VVC, and WACNN [57] are provided for reference. A smaller value is more effective.

Method	Natural Image	Comic	Line drawing	Vector art	Average
JPEG	184	447	186	676	373
BPG	33.7	88.0	28.8	114	66.2
VVC	0.00	0.00	0.00	0.00	0.00
WACNN [57]	-6.31	11.6	14.5	25.3	11.3
Yang *et al*. [53]	**-9.82**	-0.50	1.84	8.47	-0.00
Lam *et al*. [31]	151	197	161	367	219
Rozendaal *et al*. [48]	234	317	267	718	384
Zou *et al*. [56]	-9.68	-2.40	-0.13	4.12	-2.02
Ours	-9.79	**-2.82**	**-0.25**	**2.87**	**-2.50**

adaptation methods. For a fair comparison, we reimplemented Yang *et al*. [53], Rozendaal *et al*. [48], Zou *et al*. [56], and Lam *et al*. [31] in our framework. Please refer to the supplementary material for the detailed experimental setup. Additionally, we performed a comparison with the baseline method and three conventional codecs: JPEG [51], BPG [8], and VVC [10] for reference. In particular, we used VVC for intra-frames implemented in VTM [1]. We computed Bjøntegaard Delta bitrate (BD rate) [11] compared with VVC. The results are presented in Table 2. Evi-

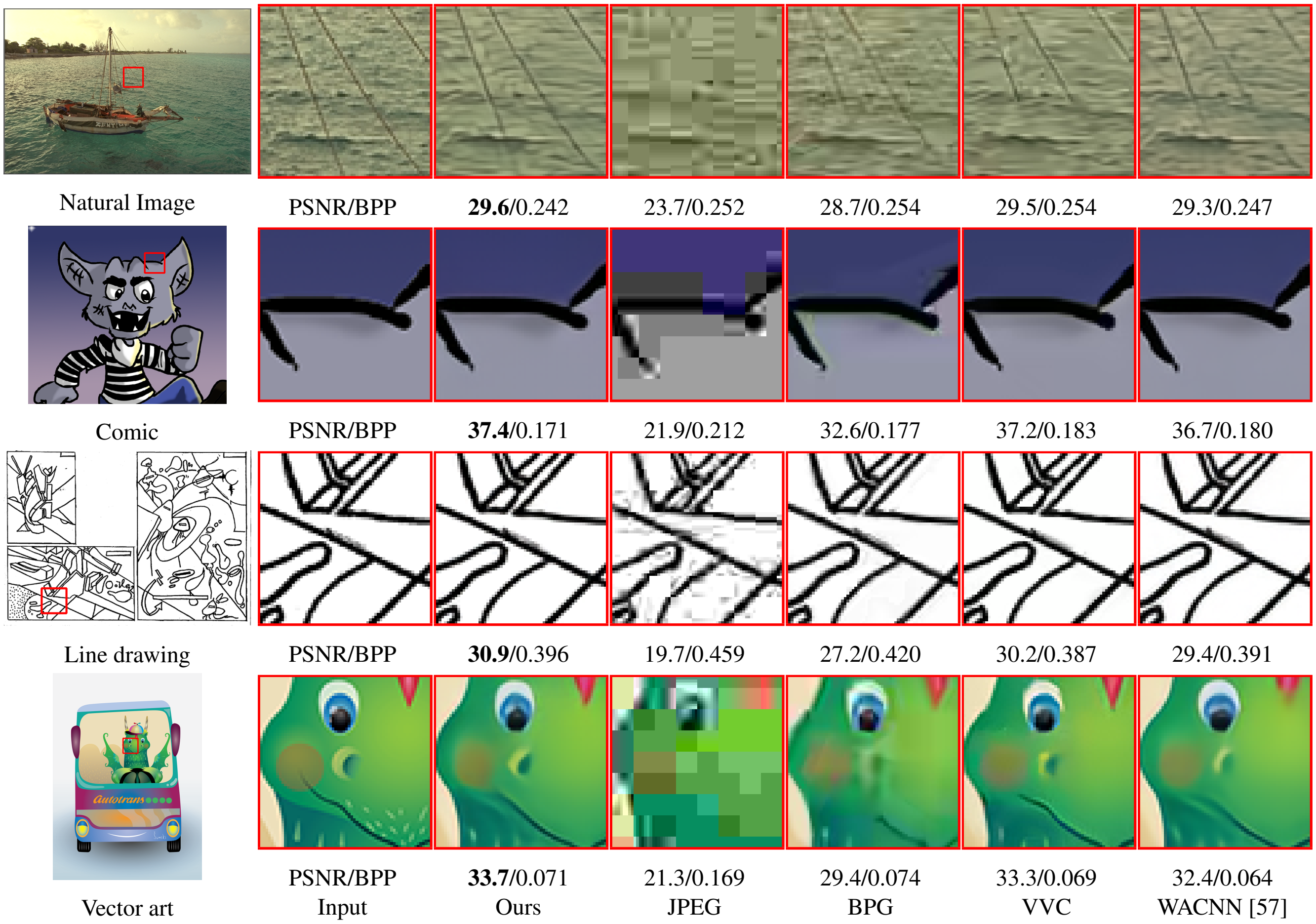


Figure 7: Qualitative results for the four domains. Ours reconstruct the wires, shadow of the character's head, overlapping lines, and the monster's nose with relatively high fidelity, respectively.

dently, the proposed method achieved performance superior to those of the other adaptation methods. Furthermore, the proposed method outperformed VVC, which is the state-of-the-art conventional method. Note that Lam *et al*. [31] and Rozendaal *et al*. [48] performed inferior to the baseline method. This is because these methods transmitted many parameters in the decoder for an individual image.

Qualitative Results. The qualitative results of the proposed method, baseline method, and conventional codecs are shown in Fig. 7. We compared these methods at a similar BPP. Our method achieved higher visual quality compared with conventional codecs and the baseline method.

Runtime. We measured the runtime of encoding and decoding using GPUs (NVIDIA Tesla V100). We conducted experiments on vector arts using the baseline and proposed methods. We show the average runtime in Table 3. The decoding time was found to be comparable to that of the baseline method. However, the proposed method required more time for encoding than the baseline method owing to the adaptive optimization framework.

Table 3: Comparison of runtime.

Method	Encoding (s)	Decoding (s)
WACNN [57]	0.16	0.16
Ours	260	0.18

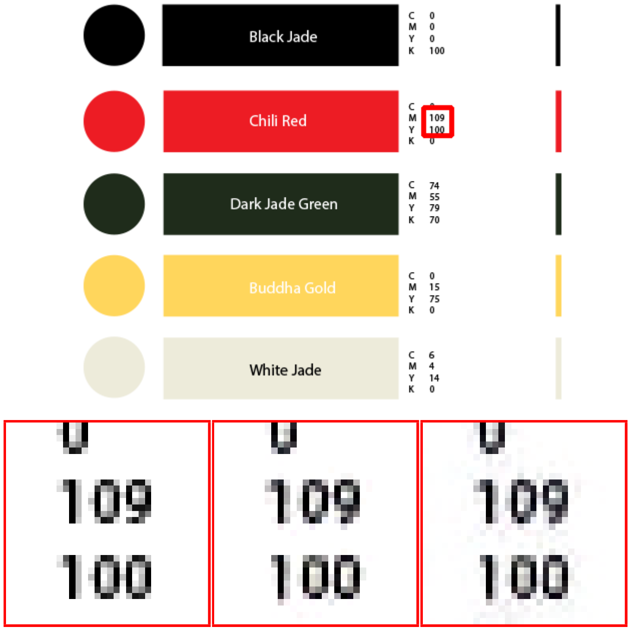
4.3. Ablation Studies

Effectiveness of Adapters. To show the effectiveness of adapters, we compared the proposed method with other methods that update parameters other than adapter parameters. In the experiments, we updated zero parameters, biases of the layers as in [31], and overfittable multiplicative parameters (OMPs) as in [56]. The numbers of updated parameters were 0, 9283, and 192, respectively.

The results on vector arts are listed in Table 4 and revealed that the highest performance is obtained when adapters are adapted. The qualitative results are shown in Fig. 8. Evidently, the artifacts around the boundary of the texts were reduced using adapters.

Table 4: Comparison of update of different parameters on BD rate (%). A smaller value is more effective.

Method	# of parameters	BD rate (%) ↓
Adapters (Ours)	768	**0.00**
Zero parameters	0	6.16
Biases	9283	42.1
OMPs	192	0.91



PSNR/BPP
Input

35.4/0.099
Ours

34.5/0.090
Without adapters

Figure 8: Qualitative results for the effectiveness of adapters. The top image is the entire input image, whereas the bottom images are the cropped patches. We can observe artifacts around the texts are reduced by using adapters.

Effectiveness of Rate-Distortion Optimization of Adapters. In our framework, we optimized the adapters in terms of rate-distortion. In this experiment, we present the results for when the adapters were optimized only in terms of distortion and compressed into eight bits as in Zou *et al.* [56]. In the implementation, we linearly transformed the parameters of the adapters to the range of $[0, 255]$ and quantized the parameters to the integer. This was performed to obtain integer values of eight bits and two real values of 32 bits, which are the scale and bias for the linear transformation. The BD rate compared with our method on vector arts was 4.21%. The results indicate the effectiveness of the rate-distortion optimization of the adapters.

Application to Another Network Architecture. Our framework can be applied to other network architectures. In this experiment, we demonstrate the performance of our framework when it is applied to Cheng20 [14]. In our implementation, we used cheng2020-attn in CompressAI [7]. We used publicly available models pre-trained on natural images. Subsequently, we inserted adapters after the convolutional layer at the first side of the final residual block of cheng2020-attn. The results are shown in Fig. 9, which revealed that our framework significantly outperformed the baseline method on Cheng20 [14].

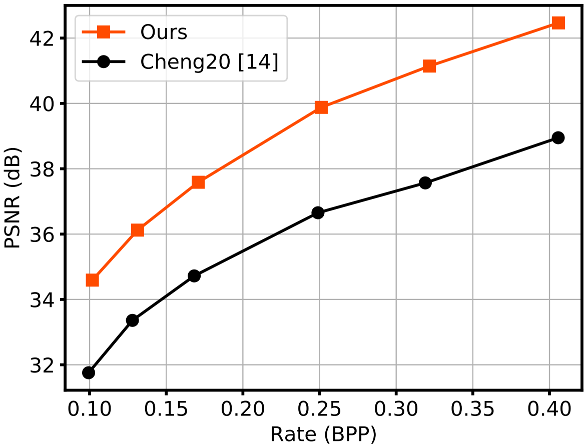


Figure 9: Results on vector arts using Cheng20 [14] as base network architecture.

Optimization Order. Our framework first optimizes the latent representation and then optimizes the parameters of the adapters. In this experiment, we swap the order of the optimization. That is, we first train the adapters and then refine the latent representation using the trained adapters. The BD rate compared with our method on vector arts was 0.70%. The results indicate that our optimization order is effective.

5. Conclusion

In this study, we addressed a novel task that we named universal deep image compression. We observed a problem wherein deep image compression deteriorates its performance on out-of-domain images. We proposed a content-adaptive optimization framework to address this problem. To adapt a pre-trained compression model per target image, we refined the latent representation extracted by the encoder and trained the adapters inserted into the decoder. Our framework can be applied to all pre-trained compression models. We constructed a benchmark dataset with four domains and demonstrated that our framework is effective. The limitation of our research is an expensive encoding time due to the optimization during compression. Reducing the encoding time is an important future work of our research.

Acknowledgement. This work was partially supported by JSPS KAKENHI Grant Number 22J13735 and 21H03460.

References

[1] Vvc official test model vtm. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tags/VTM-14.0.

[2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, pages 1122–1131, USA, July 2017.

[3] Nicola Asuni and Andrea Giachetti. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG*, pages 63–70, 2014.

[4] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density modeling of images using a generalized normalization transformation. In *ICLR*, USA, May 2016.

[5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *ICLR*, France, Apr. 2017.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, Canada, Apr. 2018.

[7] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, Nov. 2020.

[8] Fabrice Bellard. Bpg image format. https://bellard.org/bpg/.

[9] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *ACL*, pages 1–9, Dublin, Ireland, May 2022.

[10] Shan Liu Benjamin Bross, Jianle Chen and Ye-Kui Wang. Versatile video coding (draft 10). JVET-T2001, 2020.

[11] Gisle Bjøntegaard. Calculation of average psnr differences between rd-curves, 2001.

[12] Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97, pages 675–685, USA, June 2019.

[13] Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers. Content adaptive optimization for neural image compression. In *CVPRW*, USA, June 2019.

[14] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *CVPR*, pages 7936–7945, Virtual, June 2020.

[15] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *ICCV*, pages 3146–3154, Korea, Oct. 2019.

[16] Eastman Kodak Company. Kodak lossless true color image suite (photocd pcd0992). http://r0k.us/graphics/kodak/, 1993.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *NAACL*, pages 4171–4186, June 2019.

[18] Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *ACL*, pages 4884–4896, Virtual, Aug. 2021.

[19] Zongyu Guo, Zhizheng Zhang, Runsen Feng, and Zhibo Chen. Causal contextual prediction for learned image compression. *TCSVT*, 32(4):2329–2341, 2022.

[20] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *ICLR*, Virtual, Apr. 2022.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, USA, June 2016.

[22] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *COLT*, pages 5–13, USA, July 1993.

[23] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *ICML*, volume 97, pages 2790–2799, USA, June 2019.

[24] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NeurIPS*, pages 4107–4115, Spain, Dec. 2016.

[25] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *NeuIPS*, volume 34, pages 1022–1035, Virtual, Dec. 2021.

[26] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *ACL*, Virtual, Aug. 2021.

[27] Jun-Hyuk Kim, Jun-Ho Choi, Jaehyuk Chang, and Jong-Seok Lee. Efficient deep learning-based lossy image compression via asymmetric autoencoder and pruning. In *ICASSP*, pages 2063–2067, Virtual, May 2020.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, USA, May 2015.

[29] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4. *IJCV*, 128(7):1956–1981, 2020.

[30] Yat Hong Lam, Alireza Zare, Çaglar Aytekin, Francesco Cricri, Jani Lainema, Emre Aksu, and Miska M. Hannuksela. Compressing weight-updates for image artifacts removal neural networks. In *CVPRW*, June 2019.

[31] Yat Hong Lam, Alireza Zare, Francesco Cricri, Jani Lainema, and Miska M. Hannuksela. Efficient adaptation of neural network filter for video compression. In *ACMMM*, pages 358–366, Virtual, Oct. 2020.

[32] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *ICLR*, USA, May 2019.

[33] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Cross-domain few-shot learning with task-specific adapters. In *CVPR*, USA, June 2022.

[34] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, pages 4394–4402, USA, June 2018.

[35] Fabian Mentzer, George Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. In *NeurIPS*, volume 33, pages 11913–11924, Virtual, Dec. 2020.

[36] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, pages 10794–10803, Canada, Dec. 2018.

[37] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *ICIP*, pages 3339–3343, Virtual, Sep. 2020.

[38] Pramod Kaushik Mudrakarta, Mark Sandler, Andrey Zhmoginov, and Andrew G. Howard. K for the price of 1: Parameter-efficient multi-task and transfer learning. In *ICLR*, USA, May 2019.

[39] Yash Patel, Srikar Appalaraju, and R. Manmatha. Saliency driven perceptual image compression. In *WACV*, pages 227–236, Virtual, Jan. 2021.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67, 2020.

[41] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, pages 506–516, USA, Dec. 2017.

[42] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *CVPR*, pages 8119–8127, USA, June 2018.

[43] Oren Rippel and Lubomir D. Bourdev. Real-time adaptive image compression. In *ICML*, volume 70, pages 2922–2930, Australia, Aug. 2017.

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[45] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, Sep. 2001.

[46] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *CVPR*, pages 5227–5237, USA, June 2022.

[47] George Toderici, Lucas Theis, Nick Johnston, Eirikur Agustsson, Fabian Mentzer, Johannes Ballé, Wenzhe Shi, and Radu Timofte. Clic 2020: Challenge on learned image compression. `http://compression.cc`, 2020.

[48] Ties van Rozendaal, Iris AM Huijben, and Taco Cohen. Overfitting for fun and profit: Instance-adaptive data compression. In *ICLR*, Virtual, May 2021.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, USA, Dec. 2017.

[50] Chris S. Wallace. Classification by minimum-message-length inference. In *ICCI*, volume 468, pages 72–81, Canada, May 1990.

[51] G. K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, Feb. 1992.

[52] Michael J. Wilber, Chen Fang, Hailin Jin, Aaron Hertzmann, John P. Collomosse, and Serge J. Belongie. Bam! the behance artistic media dataset for recognition beyond photography. In *ICCV*, pages 1211–1220, Italy, Oct. 2017.

[53] Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. In *NeurIPS*, Virtual, Dec. 2020.

[54] Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *ICLR*, Virtual, Apr. 2022.

[55] Nannan Zou, Honglei Zhang, Francesco Cricri, Hamed R. Tavakoli, Jani Lainema, Miska Hannuksela, Emre Aksu, and Esa Rahtu. L^2C – learning to learn to compress. In *MMSP*, pages 1–6, Virtual, Sep. 2020.

[56] Nannan Zou, Honglei Zhang, Francesco Cricri, Ramin G. Youvalari, Hamed R. Tavakoli, Jani Lainema, Emre Aksu, Miska Hannuksela, and Esa Rahtu. Adaptation and attention for neural video coding. In *ISM*, pages 240–244, Italy, Nov. 2021.

[57] Renjie Zou, Chunfeng Song, and Zhaoxiang Zhang. The devil is in the details: Window-based attention for image compression. In *CVPR*, USA, June 2022.