[1] **Bin-Qian Chang**
[2] **Jar-Ferr Yang**

Deep Learned Image Compression and Semantic Segmentation Networks

Journal of Electrical Systems

Abstract: - Deep learning neural networks have become the mainstream in many computer vision tasks such as image recognition and semantic segmentation. To reduce storage space and transmission bandwidth, the images are often needed to be compressed. However, compressed images might greatly affect the performance of semantic segmentation. To balance both needs, it is necessary to optimize segmentation neural networks for specific compression algorithms with suitable compression rates. Recently, deep learning networks for image compression have been greatly improved and better than traditional coding standards. Therefore, in this paper, we propose a neural network that can adjust the compression ratio, allowing end-to-end training of image compression and semantic segmentation modules. The proposed transformer-based compression network uses the prompt tuning method to adjust the compression ratio. For semantic segmentation tasks, which take compressed image features as the input, the modified Mask2Former architecture can perform the segmentation directly. Experimental results show that the proposed method can reduce the performance loss in semantic segmentation tasks caused by image compression.

Keywords: Deep Learning, Image Compression, Semantic Segmentation, Transformer, Mask2Former.

I. INTRODUCTION

In the information explosion age, visual data are frequently transmitted through the network and stored on various storage devices. Compression techniques play an important role to save transmission bandwidth and storage spaces [1], [2]. Recently, deep learning methods have demonstrated their capabilities to dominate in many computer vision tasks. In many scenarios, we must transmit image data to the receiver, which runs the neural network for specific vision tasks. To save transmission bandwidth, we often use lossy image compression, which can cause image distortion, leading to a drop of accuracy performance of vision tasks. The traditional distortion metrics such as MSE and PSNR might not be the best way to evaluate image quality for task models.

For flying drones, we usually need the image and it segmented sensitive target objects for the observers in the receiver side. In downstream tasks in the receiver, compressed data in the receiver is typically utilized either by taking the reconstructed image as inputs [3], [4] or directly using the compressed representations as inputs [5]-[7]. If the vision task utilizes the compressed representations, we can save the decoding process, thereby to reduce the cost of computation and latency of the workflows.

To enhance the flexibility, we employ the prompt tuning method [8] to incorporate an additional quality map for adjusting the compression ratio. This feature allows us to alter the bitrate without modifying model weights. Consequently, we eliminate the need to train separate weights for varying compression ratios. In this paper, we choose semantic segmentation as the vision task and focus on finding the correlation between compression representations and task performance at different bit rates.

II. RELATED WORK

First, we review several important related work including variational autoencoder (VAE) frameworks, visual prompt tuning (VPT) and semantic segmentation in the following subsections. Combining the VAE and VPT concepts, Fig. 1 shows the original structure of the variable rate VAE-based image compression framework [15].

A. Learned Image Compression

The model architecture of the VAE-based image compression framework [9] mainly consists of an encoder and a decoder. The encoder extracts the compressed features from the pixel domain. These compressed features are then quantized and further encoded into bitstreams through arithmetic coding [10]. The bitstreams are decoded into compressed features with quantization error, and the decoder uses these features to reconstruct the image. The hyperprior helps to predict the probability distributions of the compressed representations to allow the arithmetic coding to encode the compressed representations according to these distributions. It is worth noting that to predict the distributions of compressed representations, we still need to transmit some distribution-related latent codes. The structure of the VAE-based image compression is shown in the light green box of Fig.1. In this paper, we choose a transformer-based image codec (TIC) [11] whose encoder and decoder are composed of stacks of Swin

[1] Department of Electrical Engineering, National Cheng Kung University, Taiwan. Email: chang.hchs10@gmail.com
[2] Department of Electrical Engineering, National Cheng Kung University, Taiwan. Email: jefyang@mail.ncku.edu.tw

transformer blocks (STB) [14]. For the best compression performance, the rate distortion optimization (RDO) function as,

$$L = \lambda \cdot D + R \tag{1}$$

balances the trade-off between compression rate R and distortion D, where the Lagrange multiplier λ corresponds to a fixed bitrate compression model.

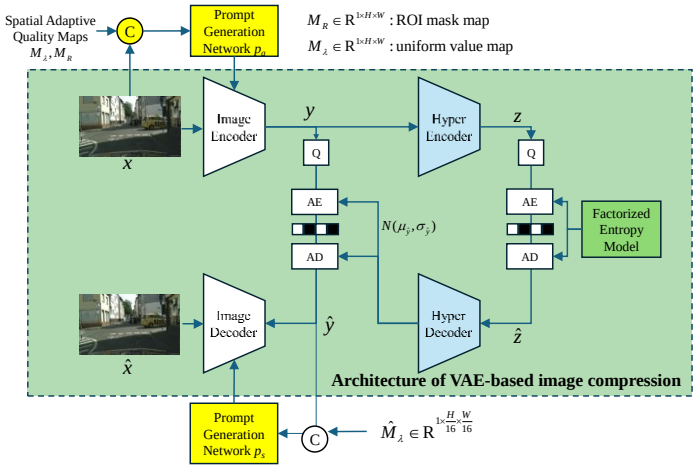

Fig. 1. Structure of the variable-rate VAE-based image compression

B. Visual Prompt Tuning

Visual prompt tuning (VPT) involves attaching additional learnable prompts to the input during training while keeping the pre-trained models fixed. Window-based multi-head self-attention with prompt tuning is shown in Fig. 2.

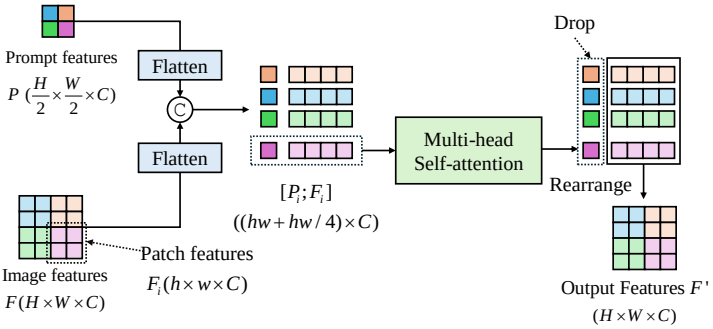

Fig. 2. Window-based multi-head self-attention with prompt tuning.

There are 4 steps in window-based attention with prompt tuning. In the 1st step, we divide the feature map $F \in R^{H\times W\times C}$ into several non-overlapping sub-feature maps $F_i \in R^{h\times w\times C}$ evenly. Similarly, we divide the prompt $P \in R^{H/2\times W/2\times C}$ into several non-overlapping sub-prompt $P_i \in R^{h/2\times w/2\times C}$ equally. In the 2nd step, we flatten the features to prepare query Q, key K and value V for multi-head self-attention. Different from original multi-head self-attention, we need to concatenate the flattened sub-feature map and sub-prompt map to become $[F_i; P_i] \in R^{(hw+hw/4)\times C}$ for key and value. Thus, the query, key and value can be respectively formulated as:

$$Q_i = F_i W_Q,\ \ K_i = [F_i; P_i] W_k \text{ and } V_i = [F_i; P_i] W_v, \tag{2}$$

where $N = h \times w$, is the total pixel number and D is the number of channels of each token with $Q_i \in R^{N\times D}$, $K_i, V_i \in R^{(N+N/4)\times D}$ and $W_q, W_k, W_v \in R^{D\times D}$. In the 3rd step, we then apply multi-head self-attention to each pair of sub-feature maps and sub-prompt. Finally, we need to drop some tokens whose positions correspond to the prompt tokens. Then, we reorganize the updated feature map by placing the remaining tokens back in their original positions.

Including the prompt into image compression, the Swin transformer block (STB) in TIC codec then contains window-based multi-head self-attention (W-MSA) and shifted window-based multi-head self-attention (SW-

MSA). Fig. 1 shows that the prompt method is included in transformer-based image compression to support variable bitrates and spatially adaptive quality control [15]. The generated spatial quality control prompts from the uniform quality map M_λ and the ROI mask M_R to control the global image quality and the ROI quality respectively. Placing the prompts in the prefix of image features before feeds into Swin transformer blocks of the image encoder. The weight coefficient λ of distortion D is a function of M_λ and M_R. The training RDO loss function can be formulated as:

$$L = 255^2 \cdot \lambda \cdot \sum_{i=1}^{N} \frac{M_{Ri} \cdot (x_i - \hat{x}_i)^2}{N} + R, \tag{3}$$

with

$$\lambda = f(M_\lambda) = \exp((\log\lambda_{\max} - \log\lambda_{\min}) \cdot M_\lambda + \log\lambda_{\min}), \tag{4}$$

where $\lambda_{\min}$ is 0.0018 and $\lambda_{\max}$ is 0.0932. Similarly, Chen *et al*. [4] applied the visual prompt tuning to optimize downstream tasks while keeping the codec frozen.

C. Semantic Segmentation

Semantic segmentation is also a crucial task in computer vision that involves partitioning an image into distinct regions based on the objects or features they contain. In this paper, we choose Mask2Former [16], a model known for its strong performance in segmentation tasks, to generate the output for our semantic segmentation task.

III. PROPOSED METHOD

The proposed semantic segmentation-oriented image compression framework shown in Fig. 3 contains TIC with a prompt generator to encode images into a variable-rate compressed domain, a transform network and a modified Mask2Former for semantic segmentation. The transform network adjusts the dimensions of the compressed domain to match the modified Mask2Former input dimensions. For aviation drone applications, for example, the proposed network can reduce the computation and the system weight burdens in the transmission side on the drone, while the image decoder and vision tasks can be performed in the receiver side on ground station without any physical limitations.

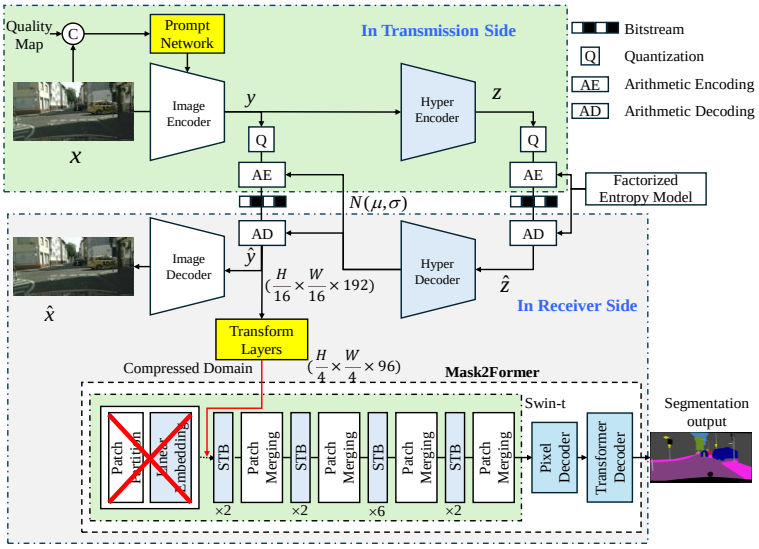

Fig. 3. Structure of the proposed semantic segmentation-oriented image compression framework.

The image compression network TIC accepts RGB images with dimensions that are multiples of 256. We concatenate the image and the quality map, feeding them into the prompt generator to guide the encoder in allocating different bitrates to the regions of interest. After encoding the image into a compressed domain guided by the quality map, the compressed representation y is passed to the hyperprior encoder, which further compresses the compressed representation into a more compact feature z with significant information loss. Here, the output dimensions of the compressed representation y are (H/16)x(W/16)x192, and the output of the hyperprior encoder z is (H/64)x(W/64)x128, where H and W denote the height and width of the input image, respectively. We quantize both representations $\hat{y}$ and $\hat{z}$. Then, we encode $\hat{z}$ into bitstreams using arithmetic coding, according to the

distribution provided by the factorized entropy model. This model collects statistical information during training and is available to both the compressor and decompressor. Next, we decode the bitstreams to obtain $\hat{z}$ via arithmetic coding and feed it into the hyperprior decoder to obtain the distribution of $\hat{y}$. Subsequently, we use this distribution to encode $\hat{y}$ and follow the same procedure to decode the bitstream into $\hat{y}$ on the decompressor side. Finally, the image decoder produces the reconstructed image from $\hat{y}$. It is not necessary to reconstruct the image $\hat{x}$ to perform the semantic segmentation task directly. To reduce the computation cost of decoding image, we use $\hat{y}$ as the input of Mask2Former for semantic segmentation. Thus, the encoded image features $\hat{y}$ can be utilized to detect objects by using Mask2Former. Of course, we can observe the reconstructed image $\hat{x}$ if we decode the encoded features $\hat{y}$.

A. Modified TransTIC

Since TransTIC performs well in many tasks, we prefer to train the prompt generator while keeping the TIC codec frozen. We can use their prompt generator to create prompts and guide the image encoder. However, their prompt generator only takes the image as the input. Thus, we modified the input dimension of the first convolution layer by adding an extra channel for quality map. Fig. 4 shows the proposed prompt generator which follows the TransTIC architecture by injecting prompts into the first two Swin transformer blocks (STB). Similar to that of Kao *et al.* [4], the proposed prompt generator does not require a lambda map for guiding bit allocation; instead, we only use a quality map.

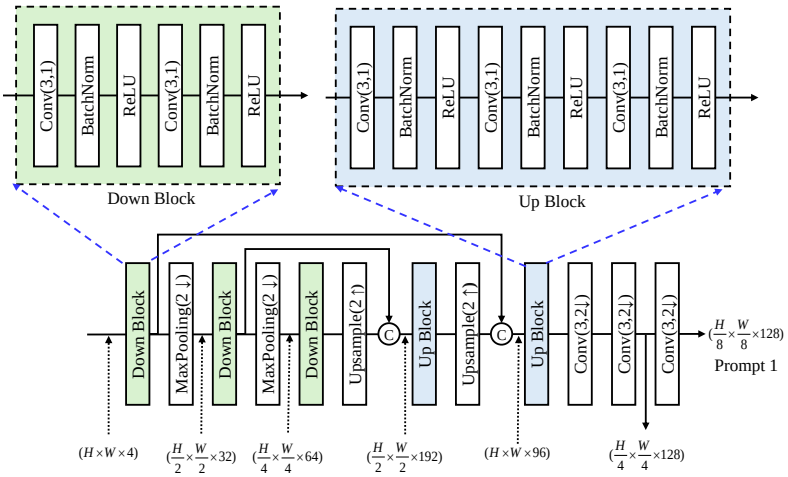

Fig. 4. Architecture of the prompt generator with down and up blocks

During training, we generate random quality maps from Perlin noise [17]. Perlin noise serves as a procedural texture primitive, a form of gradient noise utilized by visual effects artists to enhance realism in computer graphics. Perlin noise is favoured for its organic appearance, characterized by smoother transitions and a texture that closely resembles real-life textures. In the proposed network, we employ a threshold on Perlin noise to divide the quality map into two regions: the interest area and the less significant area. This quality map only consists of two distinct values: the higher one represents the region of interest, while the lower value indicates the less important area. During training, these two values are generated randomly. Examples of the quality map are shown in Fig. 5. We can observe that the boundary curve between two areas is continuous and smooth. Unlike that of Song *et al.* [18], which generates four types of quality map during training and requires segmentation annotations, our approach does not rely on segmentation annotations. This makes our method more convenient for datasets that lack these annotations.

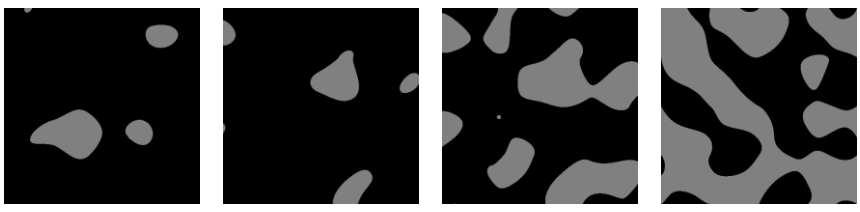

Fig. 5. Examples of Perlin noise quality map with different thresholds

B. Modified Mask2Former and Simple Transform Layer

During the semantic segmentation stage, we choose Mask2Former with Swin Transformer as the backbone to produce the semantic segmentation results. Mask2Former comprises three components: the backbone, pixel decoder, and transformer decoder. In our approach, we feed the quantized compressed features directly into Mask2Former without reconstructing them into images. To achieve this, we remove the input embedding and position embedding from Swin Transformer backbone of Mask2Former, responsible for transforming images into latent features with spatial information, while keeping the other components unchanged. By making these modifications, the input dimension of Mask2Former becomes (H/4)x(W/4)x96, which differs from the dimensions (H/16)x(W/16)x192 of the compressed representations. To align their dimensions, we introduce a simple transform layer consisting of a 1x1 convolutional layer that maps the 192 channels to 192x8 channels. Subsequently, we perform pixel shuffling to rearrange the dimensions to (H/4)x(W/4)x96, matching the input dimension of the modified Mask2Former. The detailed structure of the proposed modification is shown in Fig. 6. Utilizing the simple transform layer significantly reduces the computational cost compared to using an image decoder, resulting in smaller latency.

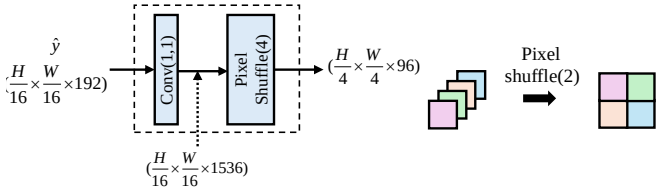

Fig. 6 Architecture of transform layer

C. Training Segmentation and Compression Framework

To achieve a variable-rate semantic segmentation-oriented compression framework, we separate the training process into four stages. The first two stages are for training variable-rate image compression networks. The last two stages are for training variable-rate semantic segmentation-oriented compression networks.

To begin with the training process, we use the TIC pre-train on Flicker2W dataset provided by TransTIC. Subsequently, the TIC is trained with a prompt generator that utilizes a uniform quality map to control the overall bitrate of the encoded image. Notably, unlike traditional prompt tuning methods that freeze the pre-trained model parameters, we train the TIC end-to-end, allowing the network to further adapt during this stage. In this stage, to minimize the rate-distortion cost, our loss function is same as (1), where the rate R and the distortion D are respectively defined as

$$R = -log_2(p_{\hat{y}}) - log_2(p_{\hat{z}}), \tag{5}$$

and

$$D = \sum_{i=1}^{N} \frac{(x_i - \hat{x}_i)^2}{N} \tag{6}$$

In (5), $p_{\hat{y}}$ and $p_{\hat{z}}$ denote the probabilities of $\hat{y}$ and $\hat{z}$ respectively. $\hat{y}$ is y with added uniform noise, i.e., $\text{uniform}(-0.5, 0.5)$ to simulate the quantization error, and $\hat{z}$ is treated similarly as $\hat{y}$. In (6), D stands for the distortion loss, where we use mean square error (MSE) as the metric to measure the distance between the original image x and the reconstructed image $\hat{x}$. λ is the Lagrange multiplier, which adjusts rate according to the values in the input quality map. It guides the model to preserve more image details when λ is high and to save more bits when λ is low. λ is defined as:

$$\lambda = \frac{1}{10}\left(\left(\frac{1}{e^4}e^{4q} - \frac{1}{e^4}\right) + \frac{1}{255^2}\right), \tag{7}$$

where $q \in [0,1]$ is the value of quality map. We do not use the loss function suggested in [15] because we want to observe the performance of semantic segmentation at much lower resolution. If we replace the λ_{min} to $(10\text{x}255^2)^{-1}$ quality control curve would be rapidly increasing when quality value is larger than 0.9. This can degrade the performance in the high-bitrate region. Therefore, we shift the exponential curve to start with $(10\text{x}255^2)^{-1}$ when q is 0.

In the second stage, we add Perlin noise quality map instead of the uniform quality map to make the encoder enhance the human perceptual quality of some interested area. Just like in the previous stage, we train the entire TIC using the rate-distortion loss. The only difference is that λ is a 2-dimensional map associated with the quality map.

The third stage focuses on training the semantic segmentation task. This is achieved by training the transform layer with an initial learning rate of 1e-4. Concurrently, the modified Mask2Former network undergoes fine-tuning with an initial learning rate of 1e-5. Importantly, the TIC and prompt generator are frozen during this stage. Since we freeze the image codec at this stage, the training does not affect the bitrate and distortion performance. The loss function L_{seg} in this stage, which is the same as that proposed for Mask2Former, is composed of classification loss L_{cls}, dice loss L_{dice}, and binary cross-entropy loss L_{mask}, respectively defined as

$$L_{\text{cls}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} g_{i,c}\log(p_{i,c}), \tag{8}$$

$$L_{\text{dice}} = 1-\frac{2}{N}\sum_{c=1}^{C}\frac{\sum_{i=1}^{N} p_{i,c}g_{i,c}+\grave{\text{o}}}{\sum_{i=1}^{N} p_{i,c}+\sum_{i=1}^{N} g_{i,c}+\grave{\text{o}}}, \tag{9}$$

$$L_{\text{mask}} = -\frac{1}{C}\frac{1}{N}\sum_{c=1}^{C}\sum_{i=1}^{N}[g_{i,c}\log(p_{i,c}) + g_{i,c}\log(p_{i,c})(1-g_{i,c})\log(1-p_{i,c})], \tag{10}$$

where N is the total number of pixels, C is the total number of classes. $p_{i,c}$ is the predicted probability for the i^{th} pixel and class c. $g_{i,c}$ is the ground truth label for the i^{th} pixel and class c. A small constant $\grave{\text{o}}$ is to ensure numerical stability. The total loss for training Mask2Former is defined as:

$$L_{\text{seg}} = 2L_{\text{cls}} + 5L_{\text{dice}} + 5L_{\text{mask}}, \tag{11}$$

In the final stage, we fine-tune the entire framework with an initial learning rate of 1e-5 except for the image encoder and decoder within the TIC model. This process aims to achieve an optimal balance between the quality of the semantic segmentation output and the bitrate of the encoded image. Unlike other approaches, we fine-tune the hyperprior portion of the image codec to achieve a lower bitrate. This modification does not alter the compressed representation itself (unless the quantization scheme is changed). Therefore, the compressed representation $\hat{y}$ can still be decoded back to the pixel domain and maintain a certain level of quality for human perception. It is important to note that if the data distribution of the dataset for the semantic segmentation task is significantly different from the training data used for pre-training the variable-rate TIC with prompt generator, we freeze the batch-normalization layer to stabilize the training process. Otherwise, the effect of the quality map for rate adjustment deviates significantly from our expectations. In this stage, we backpropagate the bitrate loss and the losses from Mask2Former to update the weights of our framework. The loss is formulated as:

$$L_{\text{total}} = R+\lambda_{\text{seg}}L_{\text{seg}}, \text{ with } \lambda_{\text{seg}} = 0.2\cdot q, \tag{12}$$

where q is the value of the quality map.

IV. Experimental Results

The proposed framework is implemented in Python 3.10.8, Pytorch 1.8.2, Torchvision 0.9.2, and CUDA 11.1. The hardware devices for the training is Intel Core i7-13700K CPU with 5.3 GHz, and NVIDIA GeForce RTX 4090 with 32GB RAM. For training the variable-rate and spatially adaptive quality control image compression network, we use the Flicker2W [19] and COCO 2017 training sets [20]. The input image size is 256×256×3 and the batch size is 8. The quality maps share the same width and height as the input image, with values ranging from 0 to 1. For training the variable-rate and semantic segmentation-oriented image compression network, we use Cityscapes dataset for both training and evaluation. The optimizer we choose is AdamW [21] with β_1 = 0.9, β_2 = 0.999, and a weight decay of 0.05, along with a polynomial learning rate scheduler. The training iteration is set to 300,000. The setting for the quality map is the same as for training the variable-rate image compression. Each quality map contains only one value. Notably, our image compression network can only handle images with heights and widths that are multiples of 256, so we must pad the images if their sizes do not meet this constraint. However, if the padded image becomes too large for the GPU memory, we must split the image into several parts,

then compress and infer them independently. This operation means the global information of the image is not shared during inference, which might cause a drop in semantic segmentation performance and require more bits to compress the image.

Cityscapes is a large-scale dataset designed for urban scene understanding with a particular focus on segmentation tasks. This dataset is widely utilized in the computer vision community for benchmarking and developing segmentation algorithms. It provides detailed annotations for various objects and classes commonly found in urban street scenes. We focus on 19 classes for the main segmentation tasks, including person, rider, car, truck, bus, train, and motorcycle, etc. For training, we utilize the training set, which comprises 2975 images. The input images are resized and cropped to 512×1024×3, and the batch size is 1. Subsequently, we validate our model on the validation set, which consists of 500 images.

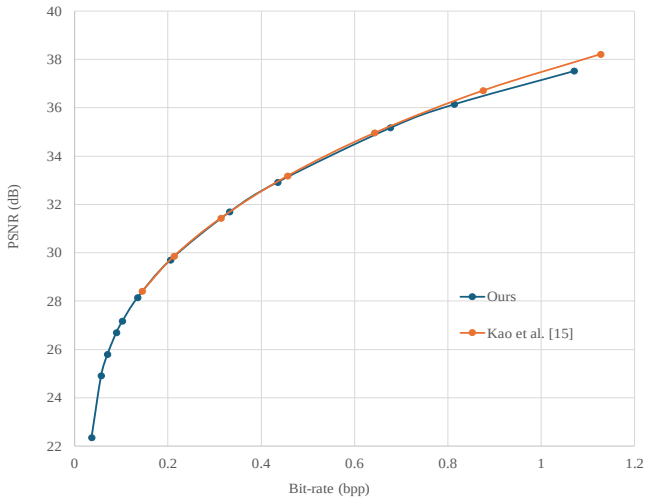

Fig. 7 PSNR comparison of the rate-distortion curve results on the Kodak dataset.

A. Results of Variable-rate Quality Control

We test the performance of our framework and compare it with Kao *et al.* [15] on Kodak dataset [22]. Fig. 7 presents the PSNR comparison of the rate-distortion curve results on the Kodak dataset. We observe that in the experiments, we can extend the rate distortion curve to a much lower bitrate region than Kao *et al.* by using (7) to control the quality rate. Our method also preserves the performance in the high bitrate region with slightly degradation near the end of the curves. We show the groundtruth and the decompressed results at different bitrates with (b) 0.105 bpp; (c) 0.226 bpp; (d) 0.439 bpp and (e) 0.763 bpp in Fig. 8.

(a) Ground Truth (b) PSNR=29.392 BPP=0,105 (c) PSNR=32.883 BPP=0.226 (d) PSNR=38.383 BPP=0.673

Fig. 8 (a) Groundtruth and the decoded results at different bitrates with (b) 0.105 bpp; (c) 0.226 bpp; (d) 0.439 bpp and (e) 0.763 bpp.

B. Results of Spatially Adaptive Quality Control

For spatially adaptive quality control, we compare our performance with Kao *et al* [15]. However, our work focuses on variable-rate semantic segmentation, and we do not use a lambda map to further adjust the rate-distortion trade-off. To ensure a fair performance comparison, we adjust the value of the non-ROI region in the ROI mask used by Kao *et al*. to achieve similar image quality in the non-ROI region. This allows us to focus on comparisons of bitrate and the ROI region. Fig. 9 shows the PSNR comparison between the ROI region and the

non-ROI region, generated based on the union of all the foreground objects in segmentation annotations of the COCO dataset. Fig. 10 shows the groundtruth image and its ROI mask while Fig. 11 exhibits our decompressed results across different spatial quality settings.

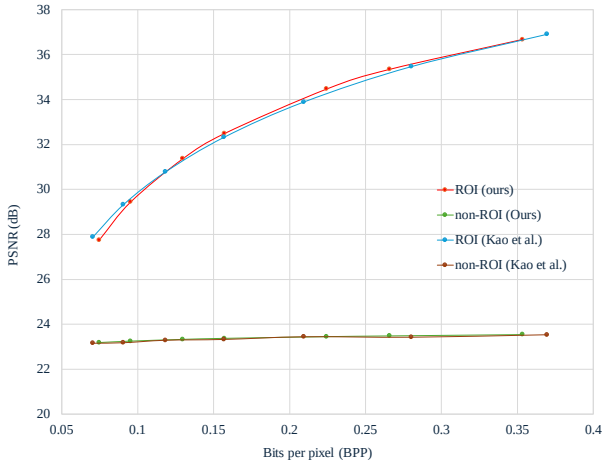

Fig. 9. PSNR comparison between the ROI region and the non-ROI region.

Ground Truth ROI Mask

Fig. 10 Ground truth mage and its ROI mask

(a) q=0.125, All PSNR:23.59 and BPP:0.076 ROI PSNR:26.06 and non-ROI PSNR:23.02
(b) q =0.35, All PSNR:24.071 and BPP:0.128 ROI PSNR:29.661 and non-ROI PSNR:23.148
(c) q =0.7, All PSNR:24.403 and BPP:0.244 ROI PSNR:33.625 and non-ROI PSNR:23.04
(d) q = 1, All PSNR:24.56 and BPP:0.382 ROI PSNR:36.019 and non-ROI PSNR:23.08

Fig. 11 Decompressed images achieved by the proposed method by setting 0 to non-ROI region in quality map and q to ROI region with (a) q=0.125; (b) q =0.35; (c) q =0.7; (d) q =1.

C. Experimental Results of Semantic Segmentation

To evaluate the segmentation performance, we calculate Intersection over Union (IoU), which is formulated as:

$$\mathrm{IoU} = \frac{TP}{TP + FP + FN} \tag{13}$$

where *TP*, *FP*, and *FN* denote true positive, false positive, and false negative rates. respectively. The IoU is defined as the ratio of the intersection area to the union area of the predicted and ground truth sets. Based on this criterion, we can then have the mean Intersection over Union (mIoU):

$$\mathrm{mIoU} = (1/C)\sum_{i=1}^{C} \mathrm{IoU}_i \,, \tag{14}$$

where C is the number of classes.

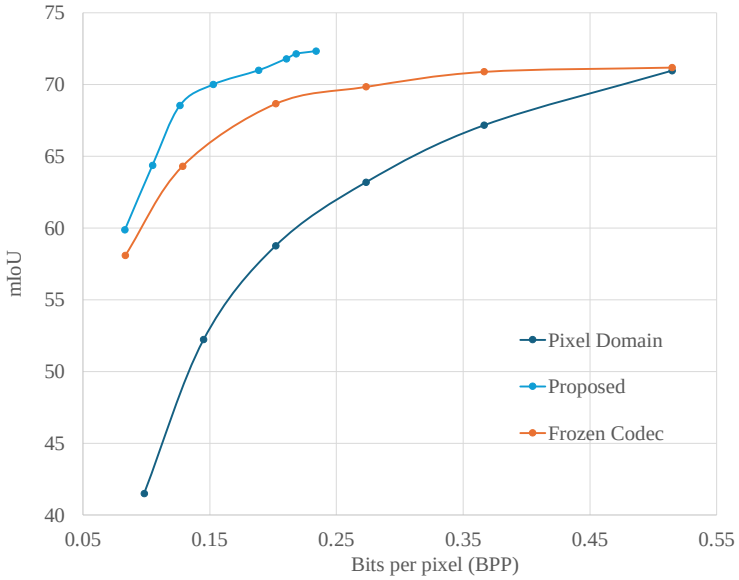

Fig. 12 Comparisons of the inference results achieved by pixel domain, Stage 3 and Stage 4 approaches, where Stage 3 without training prompt network on Cityscapes dataset.

By using the mIoU metric, we compare the detection performance of the proposed framework on the Cityscapes dataset. The experimental results characterized by mIoU-BPP curves are shown in Fig. 12. Generally, the bit per pixel (BPP) is higher, the better of mIoU will be for all methods. The proposed system improves semantic segmentation performance compared to directly feeding the reconstructed image to the original Mask2Former. We also compare the results when we freeze the image codec with the prompt generator and only train the simple transform layers and Mask2Former. Fig. 13 shows several example visualization results achieved by the proposed method tested on Cityscapes validation dataset. Table I shows the computation costs and parameters between the image decoder and transform layers when the size of input image is (512,1024,3). The proposed network only adds a transform layer with very low complexity, however, we can save the computation of the image decoder if we don't need the decoded images.

Fig. 13 Decoded images and segmentation results tested on Cityscapes validation dataset with different data rates: (a) groundtruth; (b) bpp=0.125; (c) bpp=0.162; (d) bpp=0.265

Table I. Compressions of computation cost and parameters.

Type	FLOPs(G)	Parameters (M)
Transform Layer	0.6039	0.2964
Image Decoder	97.064	2.184

V. CONCLUSIONS

In this paper, we proposed a variable-rate image compression framework to enhance the performance of semantic segmentation on compressed images. We modify the networks of the prompt generator and Mask2Former such that we can enable our framework to perform semantic segmentation at different bitrates without reloading the model weights. Additionally, by using compressed domain features, the semantic segmentation inference reduces the computational cost associated with the image decoder. In addition, we employ Perlin noise to generate the quality map during training for spatially adaptive quality control to eliminate the need for segmentation annotations. This approach is particularly convenient for datasets without such labels.

ACKNOWLEDGEMENT

This work was supported by the National Science and Technology Council under Grant NSTC 113-2221-E-006-158, Taiwan.

REFERENCES

[1] G. K. Wallace, “The JPEG still picture compression standard”, IEEE Transactions on Consumer Electronics, vol. 38, no. 1, pp. xviii–xxxiv, 1992.
[2] D. T. Lee, “JPEG 2000: Retrospective and new developments”, Proc. of the IEEE, vol. 93, no. 1, pp. 32–41, 2005.
[3] L. D. Chamain, F. Racapé, J. Bégaint, A. Pushparaja, and S. Feltman, “End-to-end optimized image compression for multiple machine tasks”, arXiv preprint arXiv:2103. 04178, 2021.
[4] Y.-H. Chen, Y.-C. Weng, C.-H. Kao, C. Chien, W.-C. Chiu, and W.-H. Peng, “TransTIC: Transferring transformer-based image compression from human perception to machine perception”, Proc. of the IEEE/CVF International Conference on Computer Vision, pp. 23297–23307, 2023.
[5] H. Choi and I. V. Bajić, “Scalable image coding for humans and machines”, IEEE Trans. on Image Processing, vol. 31, pp. 2739–2754, 2022.
[6] J. Liu, H. Sun, and J. Katto, “Learning in compressed domain for faster machine vision tasks”, Proc. of International Conference on Visual Communications and Image Processing (VCIP), pp. 01–05, 2021.
[7] J. Liu, H. Sun, and J. Katto, “Learning in compressed domain for faster machine vision tasks”, Proc. of International Conference on Visual Communications and Image Processing (VCIP), pp. 01–05, 2021.
[8] M. Jia, L Tang, BC Chen, C Cardie, S Belongie, “Visual prompt tuning”, Proc. of European Conference on Computer Vision, 2022, pp. 709–727.
[9] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior”, arXiv preprint arXiv:1802. 01436, 2018.
[10] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression”, Communications of the ACM, vol. 30, no. 6, pp. 520–540, 1987.
[11] M. Lu, P. Guo, H. Shi, C. Cao, and Z. Ma, “Transformer-based image compression”, arXiv preprint arXiv:2111. 06707, 2021.
[12] J. Liu, H. Sun, and J. Katto, “Learned image compression with mixed transformer-cnn architectures”, Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14388–14397, 2023.
[13] 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”, Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5718–5727, 2022.
[14] Z. Liu, Y Lin, Y Cao, H Hu, Y Wei, Z Zhang, S Lin, B Guo, “Swin transformer: Hierarchical vision transformer using shifted windows”, Proc. of the IEEE/CVF international conference on computer vision, pp. 10012–10022, 2021.
[15] C.-H. Kao, Y.-C. Weng, Y.-H. Chen, W.-C. Chiu, and W.-H. Peng, “Transformer-based variable-rate image compression with region-of-interest control”, in 2023 IEEE International Conference on Image Processing (ICIP), 2023, pp. 2960–2964.
[16] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, “Masked-attention mask transformer for universal image segmentation”, Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2022, pp. 1290–1299.
[17] K. Perlin, “Improving noise”, Proc. of the 29th Annual Conference on Computer Graphics and Interactive Techniques, 2002, pp. 681–682.
[18] M. Song, J. Choi, and B. Han, “Variable-rate deep image compression through spatially-adaptive feature transform”, Proc. of the IEEE/CVF International Conference on Computer Vision, pp. 2380–2389, 2021.
[19] J. Liu, G. Lu, Z. Hu, and D. Xu, “A unified end-to-end framework for efficient deep image compression”, arXiv preprint arXiv:2002. 03370, 2020.
[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, “Microsoft coco: Common objects in context”, Proc. of European Conference Computer Vision, Switzerland, September 6-12, 2014, , Part V 13, 2014, pp. 740–755.
[21] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization”, arXiv preprint arXiv:1711. 05101, 2017.
[22] Rich Franzen - Kodak, “True Color Kodak Images,”. http://r0k.us/graphics/kodak/