TransTIC: Transferring Transformer-based Image Compression from Human Perception to Machine Perception

Yi-Hsin Chen  Ying-Chieh Weng  Chia-Hao Kao  Cheng Chien  Wei-Chen Chiu  Wen-Hsiao Peng
National Yang Ming Chiao Tung University, Taiwan
{yhchen12101.cs09@, wengyc.cs09@, chiahaok.cs10@, cchien1999@cs.nycu.edu.tw}
{walon, wpeng}@cs.nctu.edu.tw

Abstract

This work aims for transferring a Transformer-based image compression codec from human perception to machine perception without fine-tuning the codec. We propose a transferable Transformer-based image compression framework, termed TransTIC. Inspired by visual prompt tuning, TransTIC adopts an instance-specific prompt generator to inject instance-specific prompts to the encoder and task-specific prompts to the decoder. Extensive experiments show that our proposed method is capable of transferring the base codec to various machine tasks and outperforms the competing methods significantly. To our best knowledge, this work is the first attempt to utilize prompting on the low-level image compression task.

1. Introduction

End-to-end learned image compression systems [8, 12, 42] have recently attracted lots of attention due to their competitive compression performance to traditional image coding methods, such as intra coding in VVC [4] and HEVC [39]. Among them, transformer-based autoencoders [45, 31, 30, 40] emerge as attractive alternatives to convolutional neural networks (CNN)-based solutions because of their high content adaptivity. Some even feature lower computational cost than CNN-based autoencoders. In common, most learned image compression systems are designed primarily for human perception.

Recently, image coding for machine perception becomes an active research area due to the rising demands for transmitting visual data across devices for high-level recognition tasks. Coding techniques in this area mainly include approaches that produce multi-task or single-task bitstreams. The methods with the multi-task bitstream feature one single compressed bitstream that is able to serve multiple downstream tasks, such as human perception and machine perception. Most of them [6, 11, 9, 27, 44] aim to learn a robust, general image representation via multi-task or contrastive learning. However, a general bitstream can hardly be rate-distortion optimal from the perspective of each individual task.

The methods with the single-task bitstream allow the image codec to be tailored for individual downstream tasks, thereby generating multiple task-specific bitstreams. One straightforward approach is to optimize a codec end-to-end for each task [5]. However, given the sheer amount of machine tasks and their recognition networks, along with the ongoing developments of new machine tasks and models, customizing a neural codec, particularly hardware-based, for every one-off machine application would be prohibitively expensive even if not impossible. Region-of-interest (ROI)-based [38] and transferring-based methods [26] are two preferred solutions. The former performs spatially adaptive coding of images according to an importance map, which can be optimized for different down-
stream tasks. The latter aims to transfer a pre-trained base codec to a new task without changing the base codec. However, how to transfer efficiently a given codec without re-training is a largely under-explored topic.

In this work, we aim to transfer a well-trained Transformer-based image codec from human perception to machine perception without fine-tuning the codec. Inspired by Visual Prompt Tuning (VPT) [17], we propose a plug-in mechanism, which injects additional learnable inputs, known as prompts, to the fixed base codec. As shown in Fig. 1(a), VPT [17] targets re-using a large-scale, pre-trained Transformer-based feature extractor on different recognition tasks. This is achieved by injecting a small amount of task-specific learnable parameters, prompts, to the Transformer-based feature extractor and learning a task-specific recognition head. Different from VPT [17], which considers only the performance of the downstream recognition task, our task focuses on image compression, which needs to strike a balance between the downstream task performance and the transmission cost (i.e., the bitrate needed to signal the bitstream). Fig. 1(b) sketches the high-level design of our proposed method. As shown, the Transformer-based encoder and decoder are initially optimized for human perception while the recognition model is an off-the-shelf recognition network. To transfer the codec from human perception to machine perception, we inject prompts to both the encoder and decoder. On the encoder side, we introduce an instance-specific prompt generator to generate instance-specific prompts by observing the input image. On the decoder side, the input image is not accessible. We thus introduce task-specific prompts to the decoder.

Our main contributions are four-fold:

- Without fine-tuning the codec, we transfer a well-trained Transformer-based image codec from human perception to machine perception by injecting instance-specific prompts to the encoder and task-specific prompts to the decoder.
- To the best of our knowledge, this work is the first attempt to utilize prompting techniques on the low-level image compression task.
- The plug-in nature of our method makes it easy to integrate any other Transformer-based image codec.
- Our proposed method is capable of transferring the codec to various machine tasks. Extensive experiments show that our method achieves better rate-accuracy performance than the other transferring-based methods on complex machine tasks.

2. Related Work

2.1. Learned Image Compression

End-to-end learned image compression systems have recently attracted lots of attention due to their competitive compression performance to traditional codecs, such as VVC [4] and HEVC [39]. Most of them adopt CNN-based autoencoders [2, 7, 8, 16, 34] with a hyperprior entropy model. Due to the great success of Transformers in high-level vision tasks, some studies [45, 31] start to explore their application to low-level vision tasks, such as image compression. In [45], Zhu et al. [45] construct an autoencoder using Swin-Transformers [29], achieving comparable compression performance to the state-of-the-art CNN-based solutions with lower computational complexity. Lu et al. [31] improves on [45] by replacing patch merging/splitting in Swin-Transformers [29] with the more general convolutional layers. Lu et al. [30] further adopt Neighborhood Attention Transformers [13] in place of Swin-Transformers for their more efficient sliding-window attention mechanism. Transformers also find applications in building efficient entropy coding models [18, 41, 33].

2.2. Compression for Machine Perception

The success of neural networks in both high-level and low-level vision tasks opens up a new research area known as compression for machine perception. That is, the compressed image features or the decoded image should be suitable for the downstream recognition tasks. We divide recent works in this emerging research area into three categories according to the characteristics of their coded bitstreams, namely, multi-task bitstreams, scalable bitstreams and single-task bitstreams.

Multi-task bitstreams. The methods [6, 11, 20] in this category generate one single compressed bitstream to serve the needs of multiple downstream tasks, such as human perception and machine perception. A straightforward approach is to train an image codec through multi-task learning [20, 6]. More recently, Feng et al. [11] utilize contrastive learning to learn a general image representation for various downstream vision tasks. An obvious disadvantage of these methods is that a single, multi-purpose bitstream can hardly be rate-distortion optimal for individual downstream tasks. Whether such a bitstream can generalize to unseen tasks is an open issue.

Scalable bitstreams. There are also approaches [9, 27, 44] that aim to generate a scalable bitstream, which can be partially decoded for simpler machine tasks or fully decoded to reconstruct the input image for human perception. However, how to arrange potentially multiple image representations for various tasks in a layered manner without introducing redundancy is challenging.

Single-task bitstreams. Different from the previous two approaches, the methods that produce single-task bitstreams allow the codec to be tailored for each individual task. A common approach is to fine-tune a pre-trained codec for a
specific downstream task [25, 32]. However, each task requires a separate model and a machine recognition task normally has a variety of recognition networks to choose from. Customizing a neural codec for each possible choice can be prohibitively expensive, particularly when the neural codec is implemented on specific hardware accelerators. In comparison, region-of-interest (ROI) coding presents a versatile coding solution. For example, Song et al. [38] propose an image codec capable of encoding an input image in a spatially adaptive way according to an importance map. The importance map can be determined at inference time to optimize the decoded image for different uses, e.g., spatial bit allocation or machine perception. Another interesting approach is presented in [26], which introduces a learnable task-specific gate module to channel-wisely selects image latents produced by a pre-trained codec for compression. Along a similar line of thinking, we propose a novel idea of transferring a pre-trained base codec to different tasks. This is achieved by introducing additional task-specific modules to adapt the base codec without changing its network weights. Specifically, we utilize prompting techniques to transfer a Transformer-based image codec from human perception to machine perception.

2.3. Prompt Tuning

The idea of prompting [21, 22, 28] was first brought up in the field of neural language processing. While transformer-based language models have been a huge success in many language tasks, fine-tuning a pre-trained, large-scale transformer model for a specific downstream task requires huge effort. Prompt tuning offers an attractive alternative, which modifies the input of text encoders while keeping their backbones untouched. Jia et al. [17] are the first to extend this approach to computer vision tasks via inserting learnable task-specific prompts to the input of the vision transformer layers. With only a small number of trainable parameters, it manages to achieve comparable or even superior performance to full fine-tuning in downstream recognition tasks.

3. Proposed Method

Given a Transformer-based image codec well-trained for human perception, i.e., the image reconstruction task, this work aims for transferring the codec to achieve image compression for machine perception (e.g., object detection) without fine-tuning the codec. We draw inspiration from [17] to propose a plug-in mechanism that utilizes prompting techniques to transfer the codec. However, unlike [17], which addresses how to adapt a pre-trained large-scale Transformer-based recognition model to different recognition tasks, our work focuses on transferring a pre-trained Transformer-based image compression model to tailor image compression for recognition tasks, an application also known as image compression for machines. As such, our performance metric involves not only the recognition accuracy but also the bit rate (in terms of bits-per-pixel) needed to signal the compressed image. In our case, the downstream recognition model can be Transformer-based or convolutional neural network-based.

In Section 3.1, we first address a classic paradigm of end-to-end learned image compression. In Section 3.2, we outline our proposed framework. This is followed by details of our transferring mechanism in Section 3.3 and the training objective in Section 3.4.

3.1. Preliminaries

An end-to-end learned image compression system usually has two major components: a main autoencoder and a hyperprior autoencoder. The main autoencoder consists of an analysis transform ($g_a$ in Fig. 2) and a synthesis transform ($g_s$ in Fig. 2). The analysis transform $g_a$ encodes an RGB image $x \in \mathbb{R}^{H \times W \times 3}$ of height $H$ and width $W$ into its latent representation $y \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times 192}$ by an encoding distribution $q_{g_a}(y|x)$. The latent $y$ is then uniformly quantized as $\hat{y}$ and entropy encoded into a bitstream by a learned prior distribution $p(\hat{y})$. On the decoder side, $\hat{y}$ is entropy decoded and reconstructed as $\hat{x} \in \mathbb{R}^{H \times W \times 3}$ through a decoding distribution $q_{g_s}(\hat{x}|\hat{y})$ realized by the synthesis transform $g_s$. In the process, the prior distribution $p(\hat{y})$ crucially affects the number of bits needed to be signal the quantized latent $\hat{y}$. It is thus modelled in a content-adaptive manner by a hyperprior autoencoder $h_a$, which comprises a hyperprior analysis transform ($h_a$ in Fig. 2) and a hyperprior synthesis transform ($h_s$ in Fig. 2). As illustrated, $h_a$ converts the image latent $y$ into the side information $z \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times 128}$, representing typically a tiny portion of the compressed bitstream. Its quantized version is decoded from the bitstream through $h_s$ to arrive at $p(\hat{y})$. Notably, this work considers a particular implementation of the main and hyperprior autoencoders, the backbones of which are Transformer-based.

3.2. System Overview

Fig. 2 illustrates our transferable Transformer-based image compression framework, termed TransTIC. It is built upon [31], except that the context prior model is replaced with a simpler Gaussian prior for entropy coding. As shown, the main autoencoder $g_a, g_s$ and the hyperprior autoencoder $h_a, h_s$ include Swin-Transformer blocks (STB) as the basic building blocks. These STB are interwoven with convolutional layers to adapt feature resolution in the data pipeline. In this work, the main and hyperprior autoencoders are pre-trained for human perception (i.e. the image reconstruction task) and their network weights are fixed during the transferring process.

To transfer $g_a, g_s$ such that the decoded image $\hat{x}$ is suitable for machine perception, we inject (1) instance-specific
prompts produced by $g_p$ into the first two STBs in $g_a$ and (2) task-specific prompts into all the STBs in $g_s$. Section 3.3 details how these additional prompts are input to these STBs. We note that the prompt generator $g_p$ and the task-specific prompts input to the decoder are learnable and updated according to the machine perception task. That is, the network weights of $g_p$ are task-specific. However, the prompts produced by $g_p$ are instance-specific because they are dependent on the input image. Section 4.4 presents ablation experiments to justify our design choices.

### 3.3. Prompting Swin-Transformer Blocks

**Swin-Transformer blocks (STB).** STB is at the very core of our design. Fig. 2 (c) details its data processing pipeline. It consists of multiple (e.g. $M$) Swin-Transformer layers (Fig. 2 (b)), with the odd-numbered layers implementing window-based multi-head self-attention (W-MSA) and the even-numbered layers realizing shifted W-MSA (SW-MSA) to facilitate cross-window information exchange. In a Swin-Transformer layer, the input is a set of tokens, each representing a feature vector at a specific spatial location. As shown, the operation of W-MSA (or SW-MSA) has four sequential steps: (1) window partition that divides evenly an input feature map $F_i \in \mathbb{R}^{h_i \times w_i \times c_i}$, $i = 1, 2, \ldots, M$, in the $i$-th Swin-Transformer layer into non-overlapping windows, (2) window flattening that flattens the feature map along the token dimension in each window, (3) multi-head self-attention that adaptively updates tokens in each window through self-attending to tokens of the same window, and (4) window collection that unflattens the updated tokens in each window and collect tokens from all the windows to form an updated feature map of dimension the same as $F_i$. In symbols, the self-attention in a window for a specific head is given by

$$
\text{Attention}(Q, K, V) = \text{Softmax}(QK^\top / \sqrt{d} + B)V,
$$

where $Q = FW_Q$, $K = FW_K$, $V = FW_V$, and $F \in \mathbb{R}^{N \times d}$ is the flattened feature map with $N$ denoting the number of tokens in the window and $d$ the dimension of each token. $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ are learnable matrices projecting the input $F$ into query $Q \in \mathbb{R}^{N \times d}$, key $K \in \mathbb{R}^{N \times d}$ and value $V \in \mathbb{R}^{N \times d}$. $B \in \mathbb{R}^{N \times N}$ is a learnable relative position bias matrix.
Transferring Encoding STBs via Instance-specific Prompting. To transfer $g_a, g_s$ to machine perception, we propose to inject additional learnable tokens, known as prompts, into the STBs. They interact with the corresponding input tokens in the pre-trained STBs to adapt the encoding process and the decoded image, in order to achieve better rate-accuracy performance for machine perception. There are two types of prompts: instance-specific prompts and task-specific prompts. Instance-specific prompts are dependent on the input image, while task-specific prompts are task dependent yet invariant to the input image. As shown in Fig. 2(a), on the encoder side, we introduce an instance-specific prompt generator $g_p$ that generates instance-specific prompts for the first two STBs (which are referred to as IP-type STBs) based on the input image. $g_p$ itself is task-specific because its network weights are trained for a specific downstream machine task. The network details of $g_p$ are provided in the supplementary document. Fig. 2(d) depicts the inner workings of IP-type STBs. They operate similarly to the ordinary STB without prompting, except that additional and separate prompts, denoted collectively as $P_i \in \mathbb{R}^{N_i \times \frac{d}{c} \times c}$, are introduced in the $i$-th Swin-Transformer layer. In particular, $P_i$ for a specific layer has a spatial resolution that is a quarter of that of $F_i$. This design choice is meant to strike a balance between compression performance and complexity. Moreover, $P_i$ is partitioned and flattened in the same way as the image feature $F_i$. In the self-attention step for a specific window and head, the prompts of the same window update the image tokens with the query $Q$, key $K$ and value $V$ matrices in Eq. (1) augmented as follows:

$$Q = FW_Q, \quad K = [F; P]W_K, \quad V = [F; P]W_V,$$

where $P \in \mathbb{R}^{N \times d}$ refers collectively to the prompts in the same window as the image tokens $F$ and $[\cdot; \cdot]$ indicates concatenation along the token dimension. With the same $W_Q, W_K, W_V$ as for Eq. (1), we have $Q \in \mathbb{R}^{N \times d}$, $K \in \mathbb{R}^{(N+\frac{d}{c}) \times d}$ and $V \in \mathbb{R}^{(N+\frac{d}{c}) \times d}$. In the window collection step, only image tokens are collected while prompt tokens are discarded.

Transferring Decoding STBs via Task-specific Prompting. Similarly, we introduce prompts to the STBs in the decoder. Unlike the encoder, the decoder adopts task-specific prompts because the input image is unavailable on the decoder side and communicating instance-specific prompts to the decoder incurs extra overhead. Specifically, these task-specific prompts are input to every STB (referred to as TP-type STB) in the decoder. Fig. 2(e) illustrates the operation of TP-type STBs. Similar to the IP-type STBs in the encoder, the TP-type STBs are prompted with separate tokens $P_i \in \mathbb{R}^{\frac{K}{c}}$ in different Swin-Transformer layers. However, within a Swin-Transformer layer, the same prompts are shared across fixed-size windows for window-based multi-head self-attention (see window flattening and multi-head self-attention). In other words, in Eq. (2) when applied to the decoder, $P$ is set to $P_i$ for different windows. This is limited by the fact that the number of fixed-size windows is variable depending on the image size. Training window- and task-specific prompts requires learning a variable number of prompts, which is infeasible.

3.4. Training Objective

In Fig. 2 the prompt generator network $g_p$ and the task-specific prompts on the decoder side are learnable while the base codec (i.e. $g_a, g_s, h_a, h_s$) is fixed. We construct the training objective in the same way as learning an image compressor. That is, the training involves minimizing a rate-distortion cost

$$L = \underbrace{- \log p(\hat{z}) - \log p(\hat{y}|\hat{z})}_{R} + \lambda \underbrace{d(x, \hat{x})}_{D},$$

where $R$ is the estimated rate needed to signal the quantized latent $\hat{y}$ and side information $\hat{z}$, $D$ characterizes the distortion between the input image $x$ and its reconstruction $\hat{x}$, and $\lambda$ is a hyper-parameter. For the sake of machine perception, the distortion measure is perceptual loss, which is evaluated with a recognition network depending on the downstream task. More details can be found in the supplementary document.

4. Experimental Results

Training Details and Datasets. We evaluate our method on three machine tasks: classification, object detection, and instance segmentation. We first follow [3] to train the base codec composed of $g_a, g_s, h_a, h_s$ on Flicker2W [24] for human perception. In this case, the distortion measure $d(\cdot, \cdot)$ in Eq. (3) is mean-squared error, and $\lambda$ is chosen to be 0.0018, 0.0035, 0.0067, and 0.013 to arrive at four separate codecs for variable-rate compression. By freezing these pre-trained base codecs, we then train the prompt generator $g_p$ together with the task-specific decoder-side prompts on ImageNet-train [10] for classification and on COCO2017-train [23] for object detection and instance segmentation. In the process, $d(\cdot, \cdot)$ in Eq. (3) is evaluated based on the perceptual loss using a pre-trained ResNet50 [15], Faster R-CNN [3] and Mask R-CNN [14] for classification, object detection and instance segmentation, respectively.

Evaluation. For classification, we use ImageNet-val [10] as the test set and a pre-trained ResNet50 [15] as the downstream recognition network. For object detection and instance segmentation, we test the competing methods on
COCO2017-val\textsuperscript{[23]} using a pre-trained Faster R-CNN\textsuperscript{[35]} and Mask R-CNN\textsuperscript{[14]} as the downstream recognition networks, respectively. Note that these recognition networks are the same as those utilized for learning $g_p$ and the decoder-side prompts. We adopt top-1 accuracy as the quality metric for classification and mean average precision (mAP) for both detection and instance segmentation.

**Baseline Methods.** The baseline methods include: (1) using the base codec ($g_a, g_s, h_a, h_s$) trained for human perception without prompting, known as TIC, (2) fine-tuning TIC end-to-end for the downstream machine tasks, i.e., full fine-tuning, (3) transferring TIC by introducing spatial feature transform (SFT) layers, termed TIC+SFT, (4) selecting partial channels of the image latent $y$ produced by TIC to perform coding for machine perception, termed TIC+channel selection, and (5) adopting region-of-interest (ROI) coding proposed in\textsuperscript{[38]}, which modulates a CNN-based compression backbone with SFT layers\textsuperscript{[43]} according to a ROI map.

For TIC+SFT, we introduce a SFT layer after every Swin-Transformer block in the encoder and decoder. In particular, the affine parameters for SFT are produced by a task-specific network that takes the coding image as input. In a sense, TIC+SFT presents an alternative to our prompting technique. For TIC+channel selection, we follow\textsuperscript{[26]} in implementing a task-specific channel selection module, which observes the image latent for channel selection, and a task-specific transform module, which converts the selected latent channels into feature maps suitable for the downstream recognition network. Both TIC+SFT and TIC+channel selection use the same base codec (i.e., TIC) and training protocol as our TransTIC to train additional task-specific networks for spatial feature transform or channel selection. More details about their implementation are provided in the supplementary document.

For ROI, we follow\textsuperscript{[38]} to extract the Grad-CAM\textsuperscript{[37]} output based on the pre-trained ResNet50\textsuperscript{[15]} as the ROI map for classification. For object detection and instance segmentation, we generate binary ROI maps according to the foreground predictions of Faster R-CNN\textsuperscript{[35]} and Mask R-CNN\textsuperscript{[14]}, respectively.

### 4.1. Rate-Accuracy Comparison

Fig. 3 visualizes the rate-accuracy plots for the competing methods. Table 1 summarizes the average bit-rate savings and accuracy improvements using the Bjontegaard metrics\textsuperscript{[51]}. BD-accuracy/mAP are computed similarly to BD-PSNR with PSNR replaced with the top-1 accuracy or mAP. Negative BD-rate numbers suggest rate saving at the same quality/accuracy level, while positive BD-PSNR/mAP/accuracy numbers suggest quality/accuracy improvements at the same bit rate. From Fig. 3 and Table 1, we make the following observations. First, (1) our TransTIC and TIC+SFT outperform TIC+channel selection across all the recognition tasks. This is attributed to the fact that both TransTIC and TIC+SFT are able to
achieve spatially adaptive coding (see Section 4.2 for their decoded images). Second, (2) our TransTIC performs comparably to TIC+SFT on the classification task and outperforms TIC+SFT on more complicated tasks, such as object detection and instance segmentation. This suggests that our prompting technique works more effectively than spatial feature transform in terms of transferring our transform-based codec. Third, (3) ROI performs the worst in between methods with spatially adaptive ability (i.e., TransTIC, TIC+SFT and ROI). It relies heavily on the quality of the ROI mask, which can later be seen in Fig. 6. Lastly, (4) full fine-tuning achieves the best performance as expected. However, this comes at the expense of having to fully fine-tune the codec for the downstream recognition model. With a wide variety of machine tasks and their recognition networks, customizing neural codecs, particularly hardware-based, for each task is impractical. One feasible approach would be to re-purpose existing neural codecs already in mass production for new machine tasks.

Fig. 4 shows that the image codec trained with full fine-tuning generalizes poorly to other unseen recognition models. Taking the classification task as an example, we replace the downstream classification model ResNet50, which is used for training under the settings of full fine-tuning and our TransTIC, with VGG19 at test time. As shown, the accuracy of full fine-tuning drops severely by 15%-20%. In contrast, our TransTIC has a less sharp decline of 3%-7%. Part of their accuracy loss comes from the smaller model capacity of VGG19 than that of ResNet50. This is evidenced by the 4%-8% accuracy loss on TIC when VGG19 is used in place of ResNet50.

To further validate the performance of our proposed method, we also compare our TransTIC with the methods recently submitted to the call-for-proposals (CFP) competition of the MPEG VCM standard based on their test protocol[1]. The results of these competing methods are from the CFP test report [36]. As shown in Fig. 5, our TransTIC performs comparably to the top performers in terms of rate-accuracy performance. However, our base codec has the additional constraint that it is optimized for human perception, while the top performers (e.g. p12, p6, p7) optimize the entire codec end-to-end for machine tasks. This shows the potential of our TransTIC.

4.2. Qualitative Results

Fig. 6 presents the decoded images and the corresponding bit allocation maps produced by the competing methods. As shown, the base codec TIC, which is optimized for human perception, tends to spend more bits on coding complex regions (e.g. the rocky surface in Fig. 6 (a) and the background forest in Fig. 6 (b)), which may be less relevant to the downstream recognition tasks. In contrast, our TransTIC and the other methods optimized for machine tasks shift more bits from the background to the foreground, resulting in generally more sharper foreground objects.

4.3. Complexity Comparison

Table 2 compares the complexity of the competing methods in terms of the kilo-multiply-accumulate-operations per pixel (kMACs/pixel) and model size. Through transferring the pre-trained TIC, our TransTIC only needs to learn for each task a prompt generation network on the encoder side
and several task-specific prompts on the decoder side. The number of these additional parameters is about one fifth of that of a complete TIC (7M). As compared with TIC, the increase in kMACs/pixel (i.e. from 188 to 202) on the decoder side is modest. In contrast, our TransTIC has a much more complex encoder than TIC because of incorporating a prompt generation network, which must adapt the encoder-side prompts to the downstream task and the current input image. Nevertheless, TransTIC offers significantly lower kMACs/pixel and parameters than ROI [38] and TIC+SFT [43], both of which utilize more computationally heavy networks for SFT layers. Furthermore, ROI [38] currently uses a pre-trained recognition network to generate the ROI mask. This results in much higher kMACs/pixel on the encoder side. Table 2 provides the kMACs/pixel needed to generate the ROI masks for different tasks. Last but not least, TIC+channel selection [26] has a low-complexity decoder (which is composed of only the hyperprior decoder and the transform module) because it need not reconstruct the input image. However, it performs much worse than our TransTIC (see Fig. 3).

### 4.4. Ablation Experiments

**IP-type vs. TP-type STBs.** This ablation experiment investigates how the prompting type, instance-specific (IP-
Table 3. Ablation variants of IP-type vs. TP-type STBs.

<table>
<thead>
<tr>
<th>Variants</th>
<th>Encoder Prompting</th>
<th>Decoder Prompting</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>IP-type</td>
<td>TP-type</td>
</tr>
<tr>
<td>B</td>
<td>TP-type</td>
<td>TP-type</td>
</tr>
<tr>
<td>C</td>
<td>IP-type</td>
<td>Disabled</td>
</tr>
<tr>
<td>D</td>
<td>IP-shared-type</td>
<td>TP-type</td>
</tr>
</tbody>
</table>

(type) or task-specific (TP-type), in the encoder and decoder STBs may impact the rate-accuracy performance. We explore four variants of the proposed method, as summarized in Table 3, where the IP-shared-type refers to instance-specific prompting with the same instance-specific prompts shared across windows in a Swin-Transformer layer. When included in the encoder STBs, these shared prompts are learned by introducing a spatially adaptive pooling layer after the prompt generation network. This ensures that the number of prompts is invariant to the image size.

In Fig. 7, we make the following observations. First, a comparison of our full model (variant A) and variant B shows that IP-type prompting works more effectively than TP-type prompting on the encoder side. Intuitively, adapting the encoding process according to the input image helps generate a bitstream better suited for the downstream recognition task in the rate-accuracy sense. Second, compared to the full model, disabling the decoder-side prompting (variants A vs. C) results in only a moderate accuracy loss, suggesting that the encoder-side prompting is more critical. Third, on the encoder side, IP-shared prompting (variant D) performs worse than IP-type prompting (variant A). That is, spatially adaptive prompting is indispensable. Last but not least, all the variants outshine TIC significantly, implying that prompting is effective in transferring TIC from human perception to machine perception.

Prompt Depth. Fig. 8 analyzes which and how many STBs to inject prompts on the encoder side. As shown, injecting prompts to early STBs closer to the input image (e.g. STB-1 with 16 prompts, and STB-1,2 with 16 prompts) is more effective than injecting them to later STBs (e.g. STB-3,4 with 16 prompts). Comparing the two early prompting variants (STB-1 with 16 prompts vs. STB-1,2 with 16 prompts), it is clear that the performance gap between them is rather limited on the easier classification task, but becomes more significant on object detection. Also, the full injection variant (STB-1,2,3,4 with 16 prompts) performs comparably to the early prompting variant (STB-1,2 with 16 prompts). We thus choose to inject prompts to STB-1,2 only.

Prompt Numbers. Fig. 8 also ablates different design choices on prompt numbers. In our design, the number of prompts in a window is chosen empirically to be 16, which is one quarter of that (i.e. 64) of the image tokens. As shown in Fig. 8, increasing the number of prompts from 16 to 64 (cp. STB-1,2 with 64 prompts vs. STB-1,2 with 16 prompts) brings little benefit on recognition accuracy. We thus stick to 16 prompts to save storage and computational cost.

5. Conclusion

This paper utilizes prompting techniques to transfer a well-trained Transformer-based image codec from human perception to machine perception. Instead of retraining the codec, we introduce additional instance-specific prompts to the Swin-Transformer layers in the encoder and task-specific prompts to the decoder. Experimental results show that our TransTIC achieves comparable or better rate-accuracy performance than the other transferring methods on various machine tasks.

Acknowledgement

This work is supported by National Science and Technology Council, Taiwan under Grants NSTC 111-2634-F-A49-010- and MOST 110-2221-E-A49-065-MY3, MediaTek, and National Center for High-performance Computing.

References


(a) Classification
(b) Object Detection

Figure 7. Ablation on IP-type and TP-type STBs.

Figure 8. Ablation on the depth and number of prompts.


