LEARNING IN COMPRESSED DOMAIN VIA KNOWLEDGE TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning in compressed domain aims to perform vision tasks directly on compressed latent representations instead of reconstructed images. Existing reports show that learning in compressed domain can achieve a comparable performance compared to that in pixel domain for certain compression models. However, we observe that when using the state-of-the-art learned compression models, the performance gap between compressed-domain and pixel-domain vision tasks is still large due to the lack of some natural inductive biases in pixel-domain convolutional neural networks. In this paper, we attempt to address this problem by transferring knowledge from pixel domain to compressed domain. We first modify neural networks for pixel-domain vision tasks to better suit compressed-domain inputs. In addition, we propose a knowledge transfer loss to narrow the gap between compressed domain and pixel domain. Experimental results on classification and instance segmentation show that the proposed method improves the accuracy of compressed-domain vision tasks significantly, which even outperforms learning on reconstructed images while avoiding the computational cost for image reconstruction.

1 INTRODUCTION

Deep neural networks (e.g., convolutional neural networks) have shown prominent performance on various computer vision tasks, such as image classification (Krizhevsky et al., 2017; He et al., 2016), object detection (Ren et al., 2016; Lin et al., 2017), and instance segmentation (He et al., 2017). In the meantime, deep learning-based image compression methods (Ballé et al., 2017; 2018; Minnen et al., 2018) also achieve better compression efficiency than conventional image coding solutions, such as JPEG (Wallace, 1992), JPEG 2000 (Skodras et al., 2001), and BPG (Bellard, 2014). However, image compression and computer vision are usually treated as two independent tasks in the research community. Generally, the original images are compressed into compact latent representations for storage and transmission, and then decompressed into RGB images before being fed into neural networks for analysis. Such a two-step process can be simplified if it is feasible to perform visual analysis on compressed latent representations directly, which saves the computing resources for image reconstruction so that analysis efficiency can be improved. Related standard activity JPEG-AI (2022) has been initiated recently, which aims to generate a single-stream, compact compressed-domain representation to support both standard image reconstruction and various computer vision tasks.

Currently, the state-of-the-art deep image compression models (Ballé et al., 2017; 2018; Minnen et al., 2018; Cheng et al., 2020; Chen et al., 2021; Qian et al., 2020) are built on the auto-encoder architecture (Hinton & Salakhutdinov, 2006), which are mainly composed of an encoder, a decoder, and an entropy model. The encoder performs a nonlinear transformation on the images and generates compressed-domain latent representations, which are quantized and transformed back into the pixel domain by the decoder. The entropy model is used to estimate the probability distribution of latents for entropy coding algorithms. In the scenario of compressed-domain computer vision analysis, the quantized latent representations are directly fed into a modified neural network for inference. The rationale behind this approach is that compressed latent representations contain enough information to reconstruct images, and thus contain enough information for vision tasks according to the data processing inequality (Cover, 1999). Therefore, learning in compressed domain can achieve at least as good performance as learning in pixel domain theoretically.

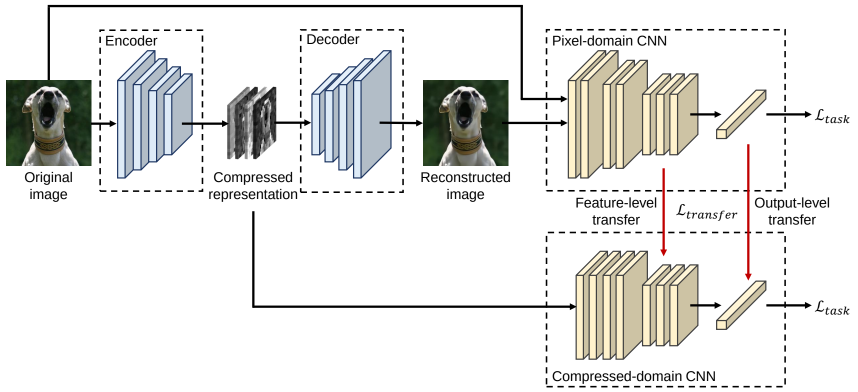


Figure 1: The proposed learning scheme for compressed-domain CNN. The knowledge transfer loss $\mathcal{L}_{transfer}$ is calculated by using the original image as input to the pixel-domain CNN.

Existing reports (Torfason et al., 2018; Wang et al., 2022) demonstrated the feasibility of learning in compressed domain. However, it is also shown in these reports that the classification accuracy from compressed domain still drops, especially for high bit rates. We also observe that when using the state-of-the-art compression models (Ballé et al., 2018; Minnen et al., 2018), the performance gap between compressed domain and pixel domain is still large on more complex tasks such as instance segmentation. We conjecture that this is because learning in compressed domain lacks the natural inductive bias of locality of convolutional neural networks (CNNs). In pixel domain, the convolutional filters with small kernel size (e.g. 3×3) extract features layer by layer, starting from pixels, with the reception field increasing progressively. Consequently, the early layers often learn local fine-grained low-level features of the input images, while the later layers learn global abstract high-level features for specific tasks. However, when learning in compressed domain, because of the downsampling operations in the encoder, each element in the compressed latents actually corresponds to a large patch of the input images. For example, the initial reception field of the compressed latents produced by a common four-layer encoder is already 61×61 pixels. Therefore, the inductive bias of locality of CNN cannot be utilized properly in the case of compressed domain, leading to worse performance.

In this paper, we propose a simple yet effective pipeline to improve the performance of compressed-domain computer vision tasks. Figure 1 shows an overview of the proposed learning scheme. Firstly, we design several variants of standard pixel-domain neural network architectures to support the use of compressed representations as input. Secondly, we propose to add a knowledge transfer loss to the original task-specific loss, which helps transfer the knowledge learned by pixel-domain CNNs to compressed-domain CNNs. To provide sufficient constraints, we consider both feature-level transfer and output-level transfer, and find that they are complementary to each other. We evaluate the proposed method on two diverse computer vision tasks including image classification and instance segmentation. For each task, we also consider the compressed latent representations from two types of image compression models (Ballé et al., 2018; Minnen et al., 2018) to validate the generalization ability of our method. The experimental results show that the performance of the compressed-domain models is improved by a large margin with the proposed transfer loss, which even outperforms the conventional models learned from reconstructed images, while enjoying lower computational complexity as image reconstruction is not needed.

2 Related Work

Learning in Compressed Domain Compressed-domain computer vision analysis is not a brand-new field of study. Some efforts focused on the compressed-domain representations from conventional codecs, e.g., the blockwise Discrete Cosine Transform (DCT) coefficients computed from JPEG codec (Wallace, 1992). Gueguen et al. (2018) demonstrated that performing image classification directly on DCT coefficients can achieve lower error rates than the baseline model with higher inference speed. Ehrlich & Davis (2019) proposed a general method for expressing convolutional networks in the JPEG domain. Xu et al. (2020) proposed to reshape the images in the frequency domain to better preserve image information in the pre-processing stage.

Recently, deep learning-based compression models have shown better performance than conventional codecs. Torfason et al. (2018) first proposed to perform image understanding on learned compressed representations. Mei et al. (2021) designed a bridge transform network to map compressed-domain features to vision task-friendly features for object detection, where the transform network is trained jointly with the compression model. Liu et al. (2021) investigated compressed-domain face alignment. Wang et al. (2022) proposed to learn from the compressed domain for original-size images. These efforts demonstrate the feasibility of learning from deep compressed representations, but the accuracy still drops significantly in some cases.

Knowledge Transfer Knowledge transfer is the process of transferring knowledge from one deep neural network (the teacher) to another (the student). When the teacher model is a larger model or an ensemble of nets, it is also known as knowledge distillation. Existing methods can be mainly divided into two types: logit-based transfer (Hinton et al., 2015; Zhao et al., 2022), and feature-based transfer (Romero et al., 2015; Heo et al., 2019; Park et al., 2019; Tian et al., 2020). Logit-based methods transfer knowledge to students by matching the softmax outputs, while feature-based methods consider matching the feature representations or the relationship between feature samples.

3 METHOD

3.1 COMPRESSED REPRESENTATION

Given an RGB image $\boldsymbol{x}$ of shape $H \times W \times 3$ and a compression model, we can get the compressed representation $\hat{\boldsymbol{y}} = Q(E(\boldsymbol{x}))$ and the reconstructed image $\hat{\boldsymbol{x}} = D(\hat{\boldsymbol{y}})$, where E, D, and Q represent encoder, decoder, and quantization, respectively. The compressed latent $\hat{\boldsymbol{y}}$ produced by learning-based compression models (Ballé et al., 2017; 2018; Minnen et al., 2018) is generally of shape $H/16 \times W/16 \times M$, where M is the number of channels and usually set as 192 or 320 according to the target bit rate. In the context of compressed-domain image understanding, the quantized latent $\hat{\boldsymbol{y}}$ is directly fed into the inference models and the decoder is not needed.

3.2 COMPRESSED-DOMAIN IMAGE CLASSIFICATION

3.2.1 MODEL ARCHITECTURE

We select ResNet-50 (He et al., 2016) as our baseline model for pixel-domain image classification. The standard ResNet-50 architecture consists of a stem block and four stages, which are named conv2, conv3, conv4, and conv5. Each stage contains several bottleneck blocks, as reported in Table 1. We also report the output size of different stages and the computational cost of the whole model when the size of input images is 256×256.

For an image of size 256×256, the size of the corresponding compressed representation is $16 \times 16 \times M$, which is equal to the output size of conv4 in the spatial dimension. Therefore, for compressed-domain ResNet, we remove the first two stages conv2 and conv3, and disable the downsampling operation in the first layer of conv4. In order to match the computational cost of the pixel-domain ResNet, we add more residual blocks in conv4 and replace the original stem block with three stacked residual blocks. See Appendix A.1 for a more detailed description of the proposed model. Then we obtain a compressed-domain ResNet architecture with similar depth and FLOPs to the pixel-domain ResNet.

3.2.2 TRAINING LOSS

For simplicity, we use superscripts to indicate the source model when referring to features or model outputs in this section and Section 3.3, where "p" is for pixel domain and "c" is for compressed domain. The pixel-domain ResNet takes image $\boldsymbol{x}$ or $\hat{\boldsymbol{x}}$ as input, and output classification probability vector $\boldsymbol{p}^p$ or $\hat{\boldsymbol{p}}^p$, as shown in Figure 1. Similarly, the compressed-domain ResNet accepts $\hat{\boldsymbol{y}}$ as input and outputs probability vector $\boldsymbol{p}^c$.

For pixel-domain classification, cross entropy loss is used by default. For compressed-domain classification, we observe that only cross entropy loss is insufficient (see Section 4). An additional transfer loss is proposed to transfer the knowledge learned from RGB images to compressed-domain

Table 1: Structure of pixel-domain and compressed-domain ResNet. The output size and FLOPs are reported for RGB images of size $256 \times 256 \times 3$ and for compressed representations of size $16 \times 16 \times 192$. The stem block and final output layer are omitted.

stage name output size	conv2_x 64×64	conv3_x 32×32	conv4_x 16×16	conv5_x 8×8	FLOPs
pixel domain	3	4	6	3	5.37×10^9
compressed domain	none	none	13	3	5.29×10^9

network. Considering the information loss in the compression process, we use the pixel-domain ResNet trained on original images $\boldsymbol{x}$ as the teacher model, while the compressed-domain ResNet is set as the student model. Inspired by the recent success of knowledge distillation techniques (Hinton et al., 2015; Romero et al., 2015; Heo et al., 2019), we consider both feature-level transfer and output-level transfer. For feature-level transfer, we extract the features $\boldsymbol{f}_{C5}$ before the final ReLU activation function of conv5 from both teacher and student models, and calculate the L_2 distance between them. For output-level transfer, Kullback–Leibler (KL) Divergence between the distributions $\boldsymbol{p}^p$ and $\boldsymbol{p}^c$ is adopted as the loss function. Therefore, the entire loss function for compressed-domain classification can be written as:

$$\begin{aligned}\mathcal{L}_{comp\_cls} &= \mathcal{L}_{task} + \lambda\mathcal{L}_{transfer} = \mathcal{L}_{CE} + \lambda(\mathcal{L}_{feature} + \mathcal{L}_{output}) \\ &= \mathcal{L}_{CE} + \lambda(D(\boldsymbol{f}^c_{C5}, \boldsymbol{f}^p_{C5}) + D_{\mathrm{KL}}(\boldsymbol{p}^p||\boldsymbol{p}^c)) \\ &= \frac{1}{N}\sum_i^N[-\sum_j^C l_{ij}\log p^c_{ij} + \lambda(||\boldsymbol{f}^c_{C5} - \boldsymbol{f}^p_{C5}||_2 + \sum_j^C p^p_{ij}\log\frac{p^p_{ij}}{p^c_{ij}})],\end{aligned} \tag{1}$$

where $\mathcal{L}_{CE}$ is the cross entropy loss between predictions and one-hot ground-truth labels $\boldsymbol{l}$, C is the number of classes, N is the number of training samples in a mini-batch, and λ is used to balance different loss terms.

3.3 Compressed-domain Instance Segmentation

3.3.1 Model Architecture

For instance segmentation, we adopt the Mask RCNN (He et al., 2017) architecture as the baseline model, which is mainly composed of a Region Proposal Network (RPN) for object bounding box proposal and an ROI head for class, box offset, and binary mask prediction. Specifically, we adopt the Feature Pyramid Network (FPN) (Lin et al., 2017) as the backbone, which uses a top-down structure with lateral connections to build a feature pyramid. When based on pixel-domain ResNet, FPN utilizes the feature maps output from all the four stages, which are denoted as $\{\boldsymbol{f}^p_{C2}, \boldsymbol{f}^p_{C3}, \boldsymbol{f}^p_{C4}, \boldsymbol{f}^p_{C5}\}$, and outputs the final feature pyramid $\{\boldsymbol{f}^p_{P2}, \boldsymbol{f}^p_{P3}, \boldsymbol{f}^p_{P4}, \boldsymbol{f}^p_{P5}, \boldsymbol{f}^p_{P6}\}$ by an iterative top-down merging process.

For compressed-domain instance segmentation, we reuse the model architecture proposed in Section 3.2.1. However, because there are no conv2 and conv3 stages, we need to modify the model structure to generate multi-scale features. Specifically, we divide the conv4 stage into three sub-stages, each containing 3, 4, and 6 bottleneck blocks, as shown in Figure 2. Then we extract features $\{\boldsymbol{f}^c_{C4-1}, \boldsymbol{f}^c_{C4-2}, \boldsymbol{f}^c_{C4}\}$ from conv4, and $\boldsymbol{f}_{C5}$ from conv5. The features $\boldsymbol{f}^c_{C4-1}$ and $\boldsymbol{f}^c_{C4-2}$ have strides of 16 pixels with respect to the input images, while the features $\boldsymbol{f}^p_{C2}$ and $\boldsymbol{f}^p_{C3}$ have strides of 4 and 8 pixels, respectively. Therefore, to match the resolution of the feature maps from pixel-domain ResNet, we utilize sub-pixel convolution (Shi et al., 2016) to upsample the features $\boldsymbol{f}^c_{C4-1}$ and $\boldsymbol{f}^c_{C4-2}$, which consists of a 1×1 convolution layer and a periodic shuffling operator. Other components such as the RPN module and the ROI head in the compressed-domain Mask RCNN keep the same as those for the pixel-domain network.

3.3.2 Training Loss

The original Mask RCNN is trained using a multi-task loss, which includes the RPN loss $\mathcal{L}_{RPN}$ for the whole image and the RoI loss $\mathcal{L}_{RoI}$ for each RoI proposal. The RPN module outputs a set

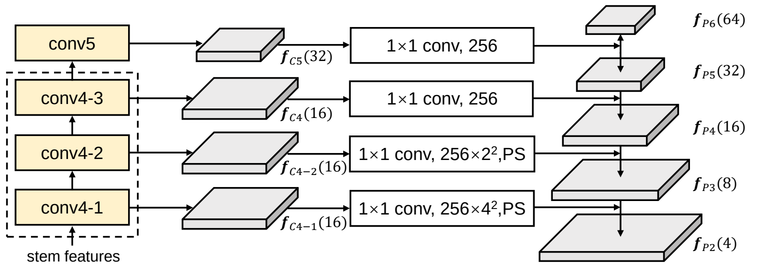


Figure 2: Illustration of the FPN structure for compressed-domain ResNet. The conv4 stage is divided into three sub-stages, i.e., conv4-1, conv4-2, and conv4-3. "1×1 conv, K" indicates a $1{\times}1$ convolution layer with K output channels. "PS" represents the periodic shuffling operator in Shi et al. (2016). The numbers in parentheses are the strides of feature maps with respect to the input image. We obtain $\boldsymbol{f}_{P6}$ by simply downsampling $\boldsymbol{f}_{P5}$ as in Lin et al. (2017).

of bounding box coordinate offsets $\boldsymbol{t}^p_{rpn}$ based on the predefined anchors, each with an objectness score $\boldsymbol{p}^p_{obj}$. For each candidate object, the RoI head outputs a K+1-dimensional class prediction $\boldsymbol{p}^p$, along with refined bounding box offsets $\boldsymbol{t}^p_{roi}$ and object masks $\boldsymbol{m}^p$ for each of the K classes.

We design the output-level transfer loss $\mathcal{L}_{output}$ based on the loss functions $\mathcal{L}_{RPN}$ and $\mathcal{L}_{RoI}$ of the pixel-domain Mask RCNN. The formulations of $\mathcal{L}_{RPN}$ and $\mathcal{L}_{RoI}$ are provided in Appendix A.2. Similar to Section 3.2.2, we use the predictions of the teacher models as soft labels, and replace the cross entropy loss with the KL Divergence loss. The ROI head of the teacher model uses the same proposal locations as the student model, while the input features are from the teacher backbone. The entire output-level transfer loss is defined as:

$$\begin{aligned}\mathcal{L}_{output} =& \mathcal{L}_{BCE}(\boldsymbol{p}^c_{obj}, \boldsymbol{p}^p_{obj}) + Smooth_{L_1}(\boldsymbol{t}^c_{rpn}, \boldsymbol{t}^p_{rpn}) \\ &+ D_{\mathrm{KL}}(\boldsymbol{p}^p, \boldsymbol{p}^c) + Smooth_{L_1}(\boldsymbol{t}^c_{roi}, \boldsymbol{t}^p_{roi}) + \mathcal{L}_{BCE}(\boldsymbol{m}^c, \boldsymbol{m}^p),\end{aligned} \tag{2}$$

where $\mathcal{L}_{BCE}$ represents the binary cross entropy loss. Note that the transfer loss is calculated on all the bounding boxes and masks predicted by the models, instead of only the positive ones, which is different from the original task loss.

The feature-level transfer loss $\mathcal{L}_{output}$ is simpler, which is defined on the feature pyramid output from the FPN backbone. L_2 distance is used to align the features between student and teacher models. The loss term is written as:

$$\mathcal{L}_{feature} = \sum_{i=2}^{6} ||\boldsymbol{f}^c_{Pi} - \boldsymbol{f}^p_{Pi}||_2. \tag{3}$$

Combining the task-specific loss and transfer loss, the entire loss function for compressed-domain instance segmentation is formulated as:

$$\mathcal{L}_{comp\_seg} = \mathcal{L}_{task} + \lambda\mathcal{L}_{transfer} = \mathcal{L}_{RPN} + \mathcal{L}_{RoI} + \lambda(\mathcal{L}_{feature} + \mathcal{L}_{output}), \tag{4}$$

where λ is a balancing weight.

4 Experiments

4.1 Learned Compression Models

We consider two state-of-the-art learned compression models (Ballé et al., 2018; Minnen et al., 2018) to generate compressed representations $\hat{\boldsymbol{y}}$ and reconstructed images $\hat{\boldsymbol{x}}$. Ballé et al. (2018) uses a hyperprior as additional side information to capture the spatial dependencies among the latents $\hat{\boldsymbol{y}}$ and the latents $\hat{\boldsymbol{y}}$ are modeled as a Gaussian distribution conditioned on the hyperprior. Minnen et al. (2018) extends the hyperprior architecture with an auto-regressive context model, which utilizes the causal context of each latent $\hat{y}_i$ to further improve the accuracy of the entropy model. We denote the compression model in Ballé et al. (2018) as *hyperprior* and that in Minnen et al. (2018) as *hyperprior-context*.

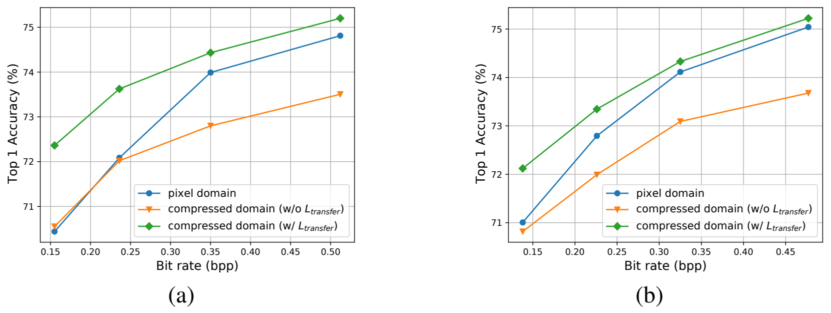


Figure 3: Classification accuracy for different input types and training losses. The compression models are (a) *hyperprior* (Ballé et al., 2018) and (b) *hyperprior-context* (Minnen et al., 2018), respectively.

In the experiments, we use the pre-trained compression models with a range of quality levels in the CompressAI (Bégaint et al., 2020) library. The parameters of the compression models are fixed during the training process of both the pixel-domain and compressed-domain models to avoid affecting the compression performance of these well-trained compression models.

4.2 Image Classfication

Dataset For classification, we evaluate our method on the ImageNet dataset (Russakovsky et al., 2015), which is a large-scale classification dataset covering 1000 classes with 1.2M training images and 50K validation images.

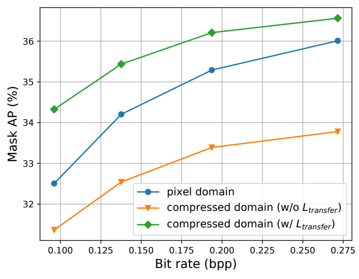
Experimental setup Because hyperprior-based compression models require the size of input images to be an integer multiple of 64, images are resized and cropped to the size of 256×256 for training and evaluation to avoid extra padding operations. For compressed-domain classification, the size of the compressed representations is $16 \times 16 \times 192$, correspondingly. For the calculation of the transfer loss for compressed-domain classification, we use the pre-trained ResNet-50 model in the PyTorch (Paszke et al., 2019) library as the teacher model. The teacher features and output logits are extracted from original images by the teacher model. The weight λ in loss function (1) is set to 1.0 by default. All the models in this experiment are trained with the same strategy for a fair comparison. Refer to Appendix A.3 for more training details.

Results We compare the classification accuracy of pixel-domain and compressed-domain models at four compression operating points. The results are shown in Figure 3. When training with only the cross entropy loss, the performance of compressed-domain models drops significantly, especially at higher bit rates. The proposed knowledge transfer loss improves the performance of compressed-domain models by a large margin at all bit rates. The average top-1 accuracy gains are 1.7 and 1.4 for *hyperprior* and *hyperprior-context*, respectively.

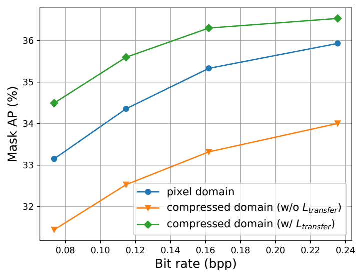
4.3 Instance Segmentation

Dataset For instance segmentation, we conduct experiments on the COCO 2017 dataset (Lin et al., 2014). The `train2017` split contains 118K images, and the `val2017` split contains 5K images.

Experimental setup Following He et al. (2017), images are resized to a maximum size of 800×1333 without changing the aspect ratio. The corresponding compressed representations have a maximum size of 50×84. The pre-trained Mask RCNN with ResNet-50-FPN backbone in the Detectron2 (Wu et al., 2019) library is used as the teacher model when training the compressed-domain models. The weight λ in loss function (4) is set to 1.0 by default. The detailed training schedule is provided in Appendix A.4.



(a)



(b)

Figure 4: Instance segmentation results for different input types and training losses. The compression models are (a) *hyperprior* (Ballé et al., 2018) and (b) *hyperprior-context* (Minnen et al., 2018), respectively.

Table 2: Computational cost for pixel-domain and compressed-domain inference. We denote the decoders of *hyperprior* and *hyperprior-context* as 'decoder$_{hp}$" and "decoder$_{hc}$", respectively.

input size	pixel domain					compressed domain
256×256	ResNet	decoder$_{hp}$	total$_{hp}$	decoder$_{hc}$	total$_{hc}$	ResNet (total)
FLOPs ($\times 10^9$)	5.37	10.01	15.38	21.57	26.94	5.29
800×1280	FPN	decoder$_{hp}$	total$_{hp}$	decoder$_{hc}$	total$_{hc}$	FPN (total)
FLOPs ($\times 10^9$)	137.15	156.41	293.56	337.05	474.2	155.25

Results The instance segmentation results for pixel-domain and compressed-domain models are shown in Figure 4. We report the standard mask average precision (AP) metric, which averages APs over IoU thresholds from 0.5 to 0.95 with an interval of 0.05. Without the transfer loss, the performance gap between compressed-domain and pixel-domain models is larger than that in classification. It is because instance segmentation requires more fine-grained pixel-level information, which is difficult to directly learn from compressed-domain representations. By transferring the semantic information learned by pixel-domain models, the performance of compressed-domain models improves notably. Our method improves the mask AP by an average of 2.9 for both compression models, and outperforms pixel-domain models by more than 1.0 mask AP on average. We also report the bounding box AP on object detection in Appendix A.5.

4.4 Computational Cost Analysis

The most important advantage of learning in compressed domain is that the computational cost of the image decoder can be saved. Table 2 reports the computational cost for pixel-domain and compressed-domain inference. For classification, given an image of size 256×256, the pixel-domain ResNet-50 requires about 5.37 GFLOPs, and the proposed compressed-domain ResNet-50 requires 5.29 GFLOPs. However, to reconstruct an image of size 256×256 from a compressed latent of size $16 \times 16 \times 192$, the computational complexity of the decoders of *hyperprior* and *hyperprior-context* are 10.01 GFLOPs and 21.57 GFLOPs, respectively, which are far more than that of inference models. Considering that real-world images to be compressed can be of much larger resolutions, learning in compressed domain is essential for a low-latency computer vision system.

For instance segmentation, the computational cost of Mask RCNN is highly dependent on the number of object candidates, which varies according to the content of images and the performance of models. Therefore, we only compare the computational cost of the FPN backbone. For an image of size 800×1280, the pixel-domain FPN backbone requires 137.15 GFLOPs, and the compressed-domain FPN backbone requires 155.25 GFLOPs. The extra complexity of compressed-domain FPN mainly comes from the sub-pixel convolutions introduced in Section 3.3.1. However, to reconstruct an image of size 800×1280, the decoders of *hyperprior* and *hyperprior-context* require 156.41 GFLOPs and 337.05 GFLOPs, respectively. Therefore, the total computational cost of the pixel-domain model is still much higher than that of the compressed-domain model.

Table 3: Ablation analysis of the transfer loss on compressed-domain classification and instance segmentation. The compression model is *hyperprior* with a quality level of 4.

λ	$\mathcal{L}_{feature}$	$\mathcal{L}_{output}$	Top 1 (%)	Top 5 (%)	bbox AP (%)	mask AP (%)
1.0			73.5	91.5	37.7	33.8
	✓		74.6	92.0	40.2	36.3
		✓	74.8	92.3	39.8	35.8
	✓	✓	**75.2**	**92.4**	**40.8**	**36.7**
0.2	✓	✓	74.4	91.8	40.5	36.4
5.0	✓	✓	75.1	92.4	40.4	36.5

4.5 Ablation Study

Type of transfer loss In this experiment, we perform knowledge transfer at feature level and output level separately to better analyze the effects of different types of knowledge transfer. The results are shown in Table 3. We observe that both losses can improve the performance of compressed-domain models individually, and it seems that they are complementary to each other. For image classification, the performance gain from output-level transfer is larger than that from feature-level transfer, while for instance segmentation, the feature-level transfer seems more important. This is reasonable because instance segmentation heavily depends on the quality of the multi-scale features to accurately locate and segment objects, while the knowledge contained in the output probability distributions is more closely associated with classification.

For classification, We also consider calculating the feature-level transfer loss on both the conv4 and conv5 features, while the output-level transfer loss keeps the same. The resulting top-1 and top-5 accuracy are 74.9 and 92.3, respectively, which are worse than using only the conv5 features.

Weight of transfer loss We adjust the weight λ in loss function (1) and (4). The results are shown in the last two rows of Table 3. We observe that the results are insensitive to the values of λ. The best performance is achieved when λ is 1.0. Finer tuning of the weights may further boost performance, but it is not the focus of this work.

Stem block We simplify the proposed compressed-domain ResNet by replacing the stacked residual blocks in the stem block with a single 1×1 convolution layer. Such a simplification reduces the computational complexity, but only has a minor impact on the accuracy. The detailed results are provided in Appendix A.6.

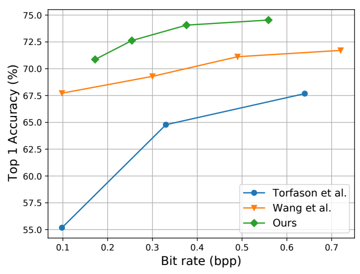
4.6 Comparison with Existing Methods

There are few methods focusing on learning in compressed domain. The most related methods are proposed by Torfason et al. (2018) and Wang et al. (2022). They designed different variants of the ResNet-50 model for compressed-domain inputs and trained the models with original task-specific loss. Following the setting in previous methods, the input images are resized and cropped to a size of 224×224 for classification. The corresponding sizes of compressed-domain representations are 28×28 for Torfason et al. (2018) and 14×14 for Wang et al. (2022) and ours.

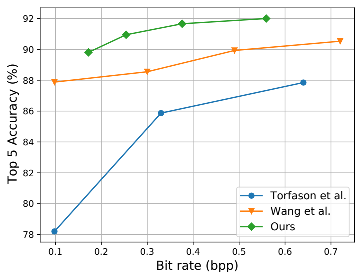
The results on ImageNet are shown in Figure 5. Our method outperforms existing methods significantly. Note that Wang et al. (2022) use the compression model in Choi et al. (2019), which is comparable to the *hyperprior* model we used. The image reconstruction quality of different methods is reported in Appendix A.7.

5 Learning in Pixel Domain via Knowledge Transfer

As shown in Figure 1, the pixel-domain CNN can use both original images and reconstructed images as input. When the model is trained on reconstructed images, the performance of vision tasks drops significantly as the quality of images degrades (see Figure 3 and Figure 4). In this section, we extend the proposed method to improve the performance of pixel-domain models. Specifically, the model trained on original images is used as the teacher model, whose knowledge is transferred to



(a)



(b)

Figure 5: Image classification accuracy on ImageNet, comparing with existing methods Torfason et al. (2018) and Wang et al. (2022). Our compression model is *hyperprior*.

Table 4: Comparison of the effects of the transfer loss on pixel-domain and compressed-domain tasks. The compression model is *hyperprior*.

	Input	$\mathcal{L}_{transfer}$	Top 1 (%)	Top 5 (%)	bbox AP (%)	mask AP (%)
Quality = 1	image		70.4	89.7	36.2	32.5
		✓	71.8	90.4	37.8	34.0
	latent		70.6	89.6	35.0	31.4
		✓	**72.4**	**90.6**	**38.2**	**34.3**
Quality = 4	image		74.8	92.1	39.8	36.0
		✓	**76.0**	**92.9**	**41.0**	**37.0**
	latent		73.5	91.5	37.7	33.8
		✓	75.2	92.4	40.8	36.7

the models trained on compressed images in the same way as in Section 3.2.2 and Section 3.3.2. We conduct experiments on both classification and instance segmentation, and use the *hyperprior* models at two compression operation points.

Table 4 shows the effect of the knowledge transfer loss on pixel-domain models, where the results in compressed domain are also provided for comparison. With the proposed transfer loss, the performance on compressed images is also improved, which illustrates that the original task loss is not sufficient for training a model on the lossy compressed images. In other words, the information contained in the compressed images cannot be fully utilized with the conventional training loss even the models have the capacity to perform better.

Another observation is that when both trained with the transfer loss, the performance of compressed-domain models is still better than that of pixel-domain models at high compression ratio (quality = 1) on both tasks, which shows another advantage of learning in compressed domain.

6 Conclusion

In this paper, we consider learning from the compressed-domain representations produced by learned state-of-the-art compression models directly. We first modify the architectures of standard ResNet and Feature Pyramid Network to support compressed-domain image classification and instance segmentation. In order to narrow the performance gap between compressed-domain and pixel-domain models, we propose to transfer the knowledge learned by pixel-domain models to compressed-domain models, which is implemented by an additional knowledge transfer loss. Experiments on ImageNet and COCO datasets demonstrate the effectiveness and generalization ability of our proposed method, which improves the performance of compressed-domain models significantly and has even superior accuracy to learning on reconstructed images. In addition, because image reconstruction is not required, our method has much lower computational complexity than learning in the pixel domain, which is crucial for a low-latency computer vision system. We also extend the proposed method to pixel-domain models, and demonstrate that it can alleviate the performance degradation caused by compression artifacts under low bit rates.

Reproducibility Statement

Our experiments are based on the open-source libraries Pytorch (Paszke et al., 2019), Compress AI (Bégaint et al., 2020) and Detectron2 (Wu et al., 2019). The experimental settings are described in Section 4.1, 4.2, and 4.3. The details of data processing steps and training process are provided in Appendix A.3 and A.4.

References

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

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*, 2020.

Fabrice Bellard. BPG image format, 2014. URL `https://bellard.org/bpg/`.

Tong Chen, Haojie Liu, Zhan Ma, Qiu Shen, Xun Cao, and Yao Wang. End-to-end learnt image compression via non-local attention optimization and improved context modeling. *IEEE Transactions on Image Processing*, 30:3179–3191, 2021.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7939–7948, 2020.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3146–3154, 2019.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Max Ehrlich and Larry S Davis. Deep residual learning in the jpeg transform domain. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3484–3493, 2019.

Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. *Advances in Neural Information Processing Systems*, 31, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1921–1930, 2019.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

JPEG-AI. Final call for proposals for jpeg ai, January 2022. SO/IEC JTC 1/SC29/WG1 N100095.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

Jinming Liu, Heming Sun, and Jiro Katto. Learning in compressed domain for faster machine vision tasks. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pp. 01–05. IEEE, 2021.

Yixin Mei, Fan Li, Li Li, and Zhu Li. Learn a compression for objection detection-vae with a bridge. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pp. 1–5. IEEE, 2021.

David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *NeurIPS*, 2018.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Yichen Qian, Zhiyu Tan, Xiuyu Sun, Ming Lin, Dongyang Li, Zhenhong Sun, Hao Li, and Rong Jin. Learning accurate entropy model with global reference for image compression. *arXiv preprint arXiv:2010.08321*, 2020.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874–1883, 2016.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

Robert Torfason, Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Towards image understanding from deep compression without decoding. In *International Conference on Learning Representations*, 2018.

Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

Zhenzhen Wang, Minghai Qin, and Yen-Kuang Chen. Learning from the cnn-based compressed domain. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3582–3590, 2022.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

Kai Xu, Minghai Qin, Fei Sun, Yuhao Wang, Yen-Kuang Chen, and Fengbo Ren. Learning in the frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1740–1749, 2020.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11953–11962, 2022.

A Appendix

A.1 Model Architecture

Table 5 shows the detailed architectures of pixel-domain and compressed-domain ResNet-50. For pixel-domain ResNet, downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2. For compressed-domain ResNet, downsampling is only performed by conv5_1. The output sizes of different stages and the FLOPs of models are reported for RGB images of size $256 \times 256 \times 3$ and for compressed latent representations of size $16 \times 16 \times 192$.

Table 5: Architectures for pixel-domain and compressed-domain ResNet-50. Building blocks are shown in brackets, with the numbers of blocks stacked.

stage name	output size	pixel domain	compressed domain
stem	64×64 (pixel) 16×16 (compressed)	$7 \times 7, 64$, stride 2 3×3 max pool, stride 2	$\begin{bmatrix} 3 \times 3, 192 \\ 3 \times 3, 192 \end{bmatrix} \times 3$
conv2_x	64×64	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	none
conv3_x	32×32	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	none
conv4_x	16×16	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 13$
conv5_x	8×8	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax	
FLOPs		5.37×10^9	5.29×10^9

A.2 Loss Functions of Mask RCNN

In this section, we provide a brief introduction of the loss functions used in Mask RCNN (He et al., 2017). For the RPN of Mask RCNN, we denote the predicted bounding box coordinate offsets as $\boldsymbol{t}^p_{rpn}$ and the predicted objectness score as $\boldsymbol{p}^p_{obj}$. Then the RPN loss is formulated as:

$$\mathcal{L}_{RPN} = \mathcal{L}_{obj}(\boldsymbol{p}^p_{obj}, \boldsymbol{p}^{p*}_{obj}) + \boldsymbol{p}^{p*}_{obj} \cdot \mathcal{L}_{reg}(\boldsymbol{t}^p_{rpn}, \boldsymbol{t}^{p*}_{rpn}), \tag{5}$$

where $\boldsymbol{p}^{p*}_{obj}$ and $\boldsymbol{t}^{p*}_{rpn}$ represent the ground-truth labels, the classification loss $\mathcal{L}_{obj}$ is the binary cross entropy loss, and the regression loss $\mathcal{L}_{reg}$ is the smooth L_1 loss, which is only calculated for positive bounding boxes.

For each candidate object, the RoI head of Mask RCNN outputs a class prediction $\boldsymbol{p}^p$, bounding box offsets $\boldsymbol{t}^p_{roi}$, and object masks $\boldsymbol{m}^p$. Then the RoI loss can be written as:

$$\mathcal{L}_{RoI} = \mathcal{L}_{cls}(\boldsymbol{p}^p, \boldsymbol{p}^{p*}) + \boldsymbol{p}^{p*} \cdot \mathcal{L}_{reg}(\boldsymbol{t}^p_{roi}, \boldsymbol{t}^{p*}_{roi}) + \boldsymbol{p}^{p*} \cdot \mathcal{L}_{mask}(\boldsymbol{m}^p, \boldsymbol{m}^{p*}), \tag{6}$$

where $\mathcal{L}_{cls}$ is the cross entropy loss, $\mathcal{L}_{reg}$ is the smooth L_1 loss, and $\mathcal{L}_{mask}$ is the average pixel-wise binary cross entropy loss. The loss terms $\mathcal{L}_{reg}$ and $\mathcal{L}_{mask}$ are also only calculated for foreground proposals, and only defined on the k-th bounding box and k-th mask for an RoI associated with ground-truth class k.

A.3 Training Details for Classification

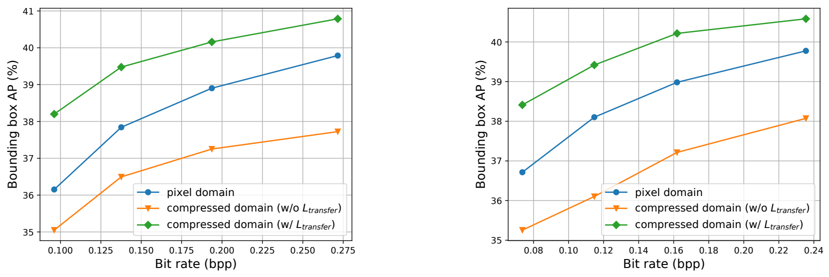
For training the classification models, images are randomly cropped and the cropped patches are resized to 256×256. Random horizontal flipping is used for data augmentation. For evaluation, images are resized first to make the shorter size 256, and then center cropping is used to generate square images of 256×256. The corresponding size of compressed representations is 16×16. We use the SGD optimizer with a batch size of 256. The learning rate starts from 0.1 and is divided by 10 every 30 epochs. The momentum of SGD is 0.9 and the weight decay is 0.0001. All the models are trained for a total of 100 epochs. We use the automatic mixed precision (amp) package in PyTorch to accelerate training.

A.4 Training Details for Instance Segmentation

The instance segmentation models are trained following the default configurations in Detectron2 (Wu et al., 2019). Images are resized with their shorter edges randomly sampled in (640, 672, 704, 736, 768, 800) for training. As for evaluation, images are resized such that the shorter edge is 800 pixels. The models are trained for 270K iterations with a batch size of 16. The SGD optimizer is applied with a weight decay of 0.0001 and a momentum of 0.9. The initial learning rate is 0.02, and is divided by 10 at 210K and 250K iterations. The automatic mixed precision technique is also used in training.

A.5 Results on Object Detection

Figure 6 shows the results on object detection, which is similar to the results on instance segmentation. By transferring the knowledge learned by pixel-domain models, the performance of compressed-domain models improves significantly. Our method improves the bounding box AP by an average of 3.0 for both compression models, and outperforms pixel-domain models by more than 1.3 box AP on average.



(a) (b)

Figure 6: Object detection results for different input types and training losses. The compression models are (a) *hyperprior* (Ballé et al., 2018) and (b) *hyperprior-context* (Minnen et al., 2018), respectively.

A.6 Ablation Study of the Stem Block

We consider replacing the original stem block in the proposed compress-domain ResNet with a simple 1×1 convolution layer. Results on classification and instance segmentation are shown in Table 6. The numbers of FLOPs are reported for compressed-domain ResNet with latents of size 16×16 as input. The simplified model has fewer FLOPs, but the accuracy for both tasks is still comparable to the original model.

Table 6: The results of compressed-domain models with different stem blocks. The compression model is *hyperprior*.

	stem block	Top 1 (%)	Top 5 (%)	bbox AP (%)	mask AP (%)	FLOPs
Quality = 1	ResBlocks	72.4	90.6	38.2	34.3	5.29×10^9
	1×1 Conv	72.0	90.6	38.1	34.3	4.91×10^9
Quality = 4	ResBlocks	75.2	92.4	40.8	36.7	5.29×10^9
	1×1 Conv	75.0	92.3	40.6	36.6	4.91×10^9

A.7 Image Reconstruction Performance

Figure 7 compares the image reconstruction performance of the compression models used in our experiments and existing methods (Torfason et al., 2018; Wang et al., 2022). We use the *hyperprior* model in Ballé et al. (2018), which is comparable to the compression model (Choi et al., 2019) used in Wang et al. (2022).

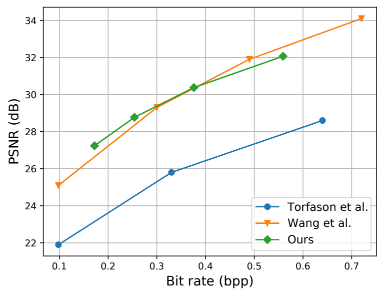


Figure 7: Image reconstruction results on ImageNet validation set, comparing with existing methods Torfason et al. (2018) and Wang et al. (2022). Our compression model is *hyperprior*.