Scalable Image Coding for Humans and Machines

Hyomin Choi, *Member, IEEE,* and Ivan V. Bajić, *Senior Member, IEEE*

Abstract—At present, and increasingly so in the future, much of the captured visual content will not be seen by humans. Instead, it will be used for automated machine vision analytics and may require occasional human viewing. Examples of such applications include traffic monitoring, visual surveillance, autonomous navigation, and industrial machine vision. To address such requirements, we develop an end-to-end learned image codec whose latent space is designed to support scalability from simpler to more complicated tasks. The simplest task is assigned to a subset of the latent space (the base layer), while more complicated tasks make use of additional subsets of the latent space, i.e., both the base and enhancement layer(s). For the experiments, we establish a 2-layer and a 3-layer model, each of which offers input reconstruction for human vision, plus machine vision task(s), and compare them with relevant benchmarks. The experiments show that our scalable codecs offer 37%–80% bitrate savings on machine vision tasks compared to best alternatives, while being comparable to state-of-the-art image codecs in terms of input reconstruction.

Index Terms—Image compression, deep neural network, multi-task network, scalable coding, latent-space scalability

I. INTRODUCTION

ADVANCES in artificial intelligence (AI) are having a major impact on both image/video coding and computer vision. New research areas are emerging at their intersection, where the goal is to develop compression systems to support both human and machine vision [1]. Related standardization activities – Video Coding for Machines (VCM) [2] and JPEG-AI [3] – have recently been initiated.

Traditionally, input compression for human vision and feature compression for machine vision have been approached separately or sequentially, through paradigms such as *compress-then-analyze* or *analyze-then-compress* [1]. Examples of the former include traditional computer vision, where it is common to train and test vision models on JPEG images, as well as compressed-domain analytics such as [4]–[11], where analysis is performed on conventionally-compressed bitstreams without full decoding or input reconstruction. Examples of the latter include Compact Descriptors for Visual Search (CDVS) [12], where machine vision-relevant features such as SIFT [13] are first extracted from the input, then compressed. In this case, however, reconstructing the input image would require significant additional bitrate.

Recent deep neural network (DNN)-based image coding methods [14]–[19] offer competitive rate-distortion (RD) performance against traditional codecs, and their perceptual performanceis even more impressive [20], [21]. DNN-based codecs map the input image into a latent space, which is then quantized and arithmetically coded [22]. Meanwhile, DNN

H. Choi and I. V. Bajić are with the School of Engineering Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada. E-mail: chyomin@sfu.ca, ibajic@ensc.sfu.ca
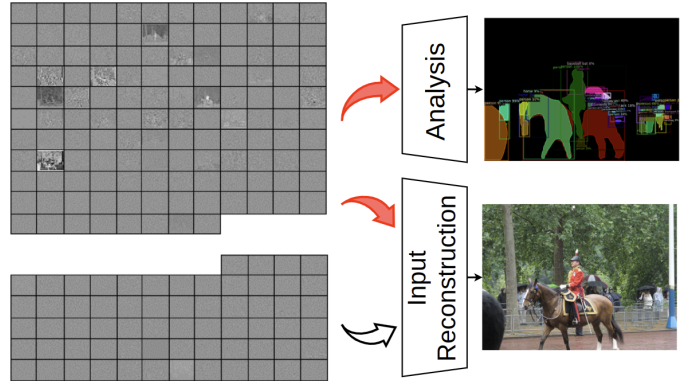

Fig. 1. An example of latent-space scalability: channels of a feature tensor (left) and the tasks they support (right).

computer vision models for classification [23], [24], object detection [25]–[27], and segmentation [28] pass the input image through a sequence of latent spaces while performing analysis. What is interesting about these latent spaces is that they are at least as compressible as the input, which we prove in the Appendix. Hence, combining image compression and DNN-based analysis is theoretically justified.

Another important trend has been the development of DNN models that support multiple tasks, including input reconstruction, using compressible representations [29]–[33]. In these methods, however, the entire latent space must be reconstructed to support any of the tasks. The most recent proposals [34]–[36] focus on multi-task scalability. For instance, [34] presented a scalable framework to support facial landmark reconstruction as the base task and input reconstruction as the enhancement task. However, both tasks rely on generative [37] decoding, which makes it hard to guarantee reconstruction fidelity. Liu *et al.* [35] present scalable image compression supporting coarse-to-fine classification as well as input reconstruction as the enhancement task. However, in this approach, multiple latent spaces are compressed, leading to inefficiency. The approach proposed in this paper compresses a single latent space, and still achieves scalability among multiple tasks, as illustrated in Fig. 1.

This paper is an extension of our recent preliminary work [36], which presented a 2-layer model based on the YOLOv3 [27] backbone to support object detection and input reconstruction. In [36], the enhancement portion of the latent space did not need to be reconstructed, but it still needed to be entropy decoded. Those initial ideas are extended in the present paper in the following ways:

- We extend the 2-task model from [36] so that base and enhancement layers are now in separate bitstreams, and
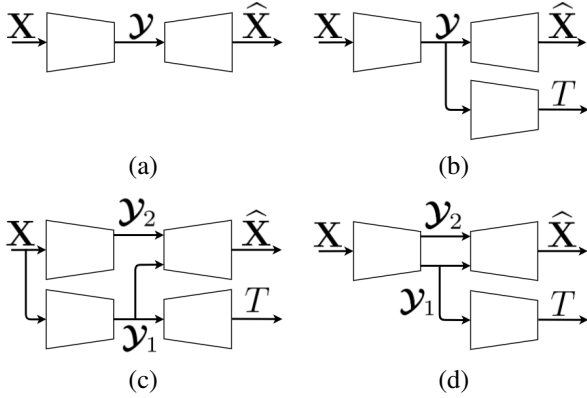
Fig. 2. Various frameworks of DNN-based compression system. (a) Input compression-only. (b) Multi-task with a single bitstream. (c) Multi-task with separately optimized scalable bitstream. (d) Multi-task with jointly optimized scalable bitstream with latent-space scalability.

the enhancement layer does not need to be decoded to support the base task.

- We develop a 3-layer model based on a Feature Pyramid Network (FPN) [38] backbone, which supports object detection, object segmentation, and input reconstruction.
- We provide theoretical analysis of latent-space compressibility (Appendix).
- We provide additional analysis and mutual information measurements showing latent-space task-specific information steering facilitated by the proposed loss function.
- We provide a more extensive experimental validation, including an ablation study and comparison with a wider range of benchmarks.

In Section II, we present preliminaries related to DNN-based multi-task image compression. Section III presents the proposed method for multi-task image coding with latent-space scalability. Experimental results are presented in Section IV, followed by conclusions in Section V.

II. PRELIMINARIES AND RELATED WORK

DNN-based image compression approaches [14]–[18] map the input image $\mathbf{X}$ into a latent representation $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{N \times M \times C}$, as shown in Fig. 2(a), where C is the number of latent feature channels and $N \times M$ is the resolution of each channel. The latent representation $\boldsymbol{\mathcal{Y}}$ is compressed, and later decoded to produce the reconstructed input $\widehat{\mathbf{X}} \approx \mathbf{X}$. Several multi-task DNNs [29]–[31] are based on a similar approach, but in addition to input reconstruction $\widehat{\mathbf{X}}$, they enable computer vision inference T (e.g., classification, object detection, etc.) from the latent space without input reconstruction, as shown in Fig. 2(b). However, with these approaches, the entire latent representation $\boldsymbol{\mathcal{Y}}$ needs to be reconstructed in order to produce either $\widehat{\mathbf{X}}$ or T, which is inefficient, as will be discussed in Section III-A. In particular, it will be seen that the information needed for T is a subset of that needed for $\widehat{\mathbf{X}}$, which is the main motivation behind our proposed latent-space scalability.

Huo *et al.* in [34] proposed a scalable multi-task compression framework for face images, whose structure is shown

in Fig. 2(c). Here, $\boldsymbol{\mathcal{Y}}_1$ is the edge map, and $\boldsymbol{\mathcal{Y}}_2$ is color information. $\boldsymbol{\mathcal{Y}}_1$ is used for face landmark detection T, while both $\boldsymbol{\mathcal{Y}}_1$ and $\boldsymbol{\mathcal{Y}}_2$ are used for reconstructing the input face image $\widehat{\mathbf{X}}$. While this approach gains efficiency from scalability (the base layer $\boldsymbol{\mathcal{Y}}_1$ is used for both task), its main drawbacks are that it is specific to face images, and uses generative decoding, which makes $\widehat{\mathbf{X}}$ plausible, but difficult to guarantee fidelity of input reconstruction ($\widehat{\mathbf{X}} \approx \mathbf{X}$). Most recently, Liu *et al.* [35] proposed "semantic-to-signal" scalable image compression, which can support multiple classification tasks in a coarse-to-fine manner, but requires coding and reconstruction of multiple latent spaces.

The approach proposed in this paper, whose preliminary version is described in our recent work [36], is illustrated in Fig. 2(d) (and in more detail in Fig. 1). A single latent space is split into two parts, $\{\boldsymbol{\mathcal{Y}}_1, \boldsymbol{\mathcal{Y}}_2\}$, where $\boldsymbol{\mathcal{Y}}_1$ represents the base layer and $\boldsymbol{\mathcal{Y}}_2$ is the enhancement layer. Only $\boldsymbol{\mathcal{Y}}_1$ needs to be decoded to perform inference task T, whereas both $\{\boldsymbol{\mathcal{Y}}_1, \boldsymbol{\mathcal{Y}}_2\}$ are used to reconstruct $\widehat{\mathbf{X}}$. The approach is generic and able to support multiple inference tasks T, which we demonstrate by constructing and testing a 3-layer system later in the paper.

III. PROPOSED METHODS

A. The backbone

In developing a scalable DNN-based multi-task compression system, the first step is to choose the backbone that can support the tasks of interest. Since one of the tasks is input reconstruction, we select a recent backbone from [16], which has shown competitive performance against conventional image codecs. Its operation can be described as [14]–[18]:

$$\begin{aligned}
\boldsymbol{\mathcal{Y}} &= g_a(\mathbf{X}; \phi) \\
\widehat{\boldsymbol{\mathcal{Y}}} &= Q(\boldsymbol{\mathcal{Y}}) \\
\widehat{\mathbf{X}} &= g_s(\widehat{\boldsymbol{\mathcal{Y}}}; \theta)
\end{aligned} \tag{1}$$

where g_a and g_s are the analysis and synthesis transforms, respectively, ϕ and θ are their parameters, Q represents quantization, and the lossless operations of entropy coding and decoding have been omitted from (1).

Let T be a machine vision inference task that needs to be derived from the input image $\mathbf{X}$. Since $\widehat{\mathbf{X}} \approx \mathbf{X}$ at high enough
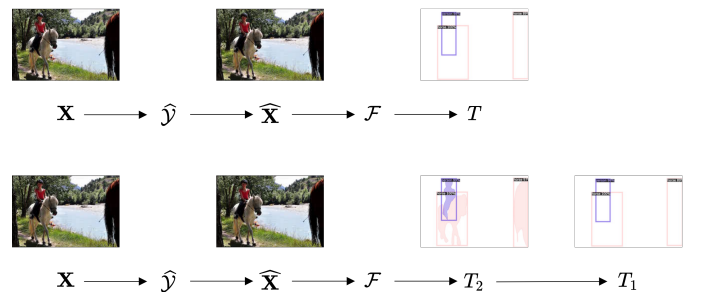

Fig. 3. Illustration of the data processing inequality for the 2-task model (top) and the 3-task model (bottom).
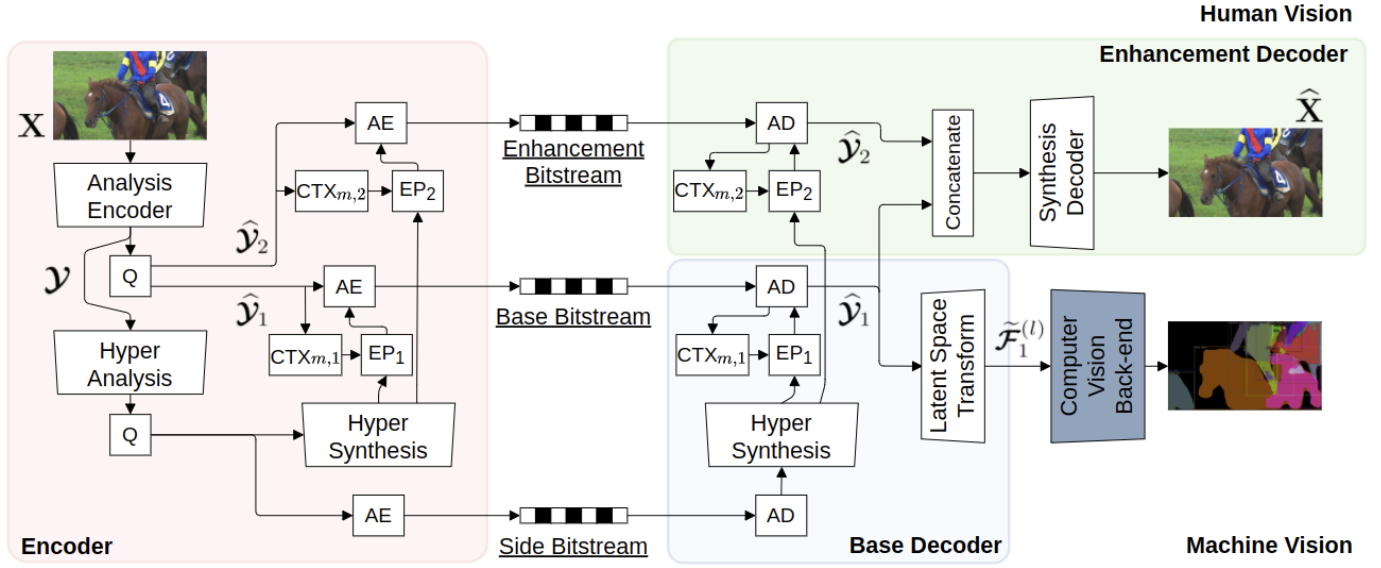
Fig. 4. An example 2-layer embodiment of the proposed scalable image coding approach. AE/AD represent arithmetic encoder and decoder, respectively. CTX and EP stand for context model and entropy parameters, originally all borrowed from Minnen *et al.* [15].

rate, this inference can also be derived from $\widehat{\mathbf{X}}$.[1] Let V be a DNN that computes T, which operates as:

$$
\begin{aligned}
\mathcal{F} &= V^{(\text{front-end})}(\widehat{\mathbf{X}}; \psi) \\
T &= V^{(\text{back-end})}(\mathcal{F}; \rho)
\end{aligned}
\tag{2}
$$

where ψ and ρ are the parameters of the DNN's front-end and back-end, respectively. $V^{(\text{front-end})}$ consists of the DNN's input layer and a number of subsequent layers, and generates the intermediate feature tensor $\mathcal{F}$. $V^{(\text{back-end})}$ consists of the remaining DNN layers and produces the inference output T.

The processing pipeline described by (1) and (2) forms a Markov chain $\mathbf{X} \to \widehat{\mathcal{Y}} \to \widehat{\mathbf{X}} \to \mathcal{F} \to T$. This chain is illustrated in Fig. 3 (top), where the inference task T is object detection. Applying the data processing inequality (DPI) [39] to the Markov chain, we get

$$
I(\widehat{\mathcal{Y}}; \widehat{\mathbf{X}}) \geq I(\widehat{\mathcal{Y}}; \mathcal{F}) \geq I(\widehat{\mathcal{Y}}; T),
\tag{3}
$$

where $I(\cdot; \cdot)$ denotes mutual information [39]. We can draw two conclusions from this analysis. First, the latent representation $\widehat{\mathcal{Y}}$ contains enough information to compute T, because it contains enough information to reconstruct $\widehat{\mathbf{X}}$, from which T can be computed. Second, $\widehat{\mathcal{Y}}$ carries less information about T than it does about $\widehat{\mathbf{X}}$. This motivates our approach to latent-space scalability - we construct $\widehat{\mathcal{Y}}$ in such a way that only part of it is used to compute T, while all of it is used to reconstruct $\widehat{\mathbf{X}}$. Note that the latent space $\widehat{\mathcal{Y}}$ is not the same as $\mathcal{F}$ (the space from which $V^{(\text{back-end})}$ computes T), so we will also need to construct the latent space transform from $\widehat{\mathcal{Y}}$ to $\mathcal{F}$ in order to bypass $\widehat{\mathbf{X}}$ when only T is needed. Details of the latent space transforms will be discussed in Section III-C.

We will present two embodiments of the above idea: a 2-task (2-layer) system, and a 3-task (3-layer) system. The 2-

layer system supports object detection from the base layer and input reconstruction from the full latent space, as explained above. The 3-layer system supports object detection (T_1), object segmentation (T_2), and input reconstruction ($\widehat{\mathbf{X}}$). The processing chain for such a system can be represented as $\mathbf{X} \to \widehat{\mathcal{Y}} \to \widehat{\mathbf{X}} \to \mathcal{F} \to T_2 \to T_1$, and is illustrated in Fig. 3 (bottom). Note that it is possible to derive object detection results (T_1) from segmentation results (T_2) by simply placing bounding boxes around segmented objects and copying the object class. This implies $I(\widehat{\mathcal{Y}}; T_2) \geq I(\widehat{\mathcal{Y}}; T_1)$, and moreover, information required for object detection (T_1) is a subset of that required for segmentation (T_2). For this reason, in our 3-layer system, object detection is placed in base layer, segmentation is supported by the base and the first enhancement layer, while the whole latent space supports input reconstruction.

The processing chains in Fig. 3 are shown only to illustrate the DPI, and explain the resulting design of the latent space(s) in our systems. In practice, there is no need to reconstruct the input image $\widehat{\mathbf{X}}$ if one wants only to perform object detection or segmentation. This can be achieved directly from the corresponding portion of the latent space through the appropriate latent space transform, as discussed in Section III-C.

B. Task-scalable image coding

Fig. 4 shows an embodiment of our 2-task (2-layer) system. In particular, we adopt the baseline network [16] using mean and scale hyperprior to capture spatial dependencies in $\mathcal{Y}$, estimated by entropy parameter (EP) module and context model (CTX) for arithmetic encoder/decoder (AE/AD). Also, we adopt the configurations for Analysis Encoder, Synthesis Decoder, and Hyper Analysis/Synthesis without attention layers from [16]. The Analysis Encoder transforms the input image $\mathbf{X}$ into $\mathcal{Y} \in \mathbb{R}^{N \times M \times C}$, with $C = 192$ as in [15], [16].

---

[1]In fact, this is common in practice: computer vision models are usually trained and used on JPEG images, not raw images.
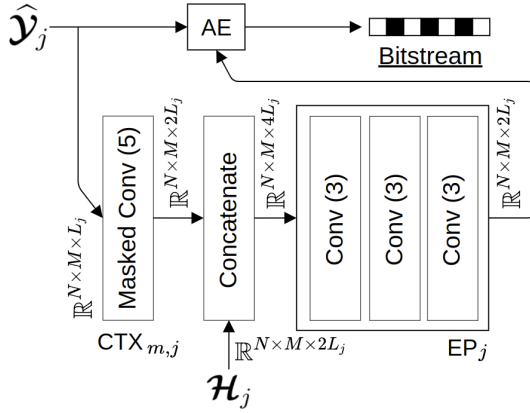
Fig. 5. Entropy coding-related blocks with tensor dimension specification.

Fig. 6. Architecture of the latent space transform (LST) block

The latent representation $\mathcal{Y}$ is split into sub-latents $\{\mathcal{Y}_1, \mathcal{Y}_2\}$, where $\mathcal{Y}_1 = \{\mathbf{Y}_1, \mathbf{Y}_2, ... \mathbf{Y}_i\}$ represents the base layer and $\mathcal{Y}_2 = \{\mathbf{Y}_{i+1}, \mathbf{Y}_{i+2}, ..., \mathbf{Y}_C\}$ represents the enhancement layer. The quantized sub-latent $\widehat{\mathcal{Y}}_j$ is coded using its own context model ($\text{CTX}_{m,j}$) and entropy parameter module (EP_j). The side bitstream, which contains coded hyper-parameters, is used by both layers.

Details of entropy coding related-blocks along with dimensions of relevant tensors are presented in Fig. 5. Let $\widehat{\mathcal{Y}}_j \in \mathbb{R}^{N \times M \times L_j}$ be the j-th quantized sub-latent, where L_j is the number of channels in the j-th sub-latent. $\text{CTX}_{m,j}$ is the context model for $\widehat{\mathcal{Y}}_j$ associated with a masked convolutional layer [15], [16] with 5×5 kernels (Masked Conv (5) in the figure), to produce an output tensor with $2L_j$ channels. This output tensor is concatenated with $\mathcal{H}_j \in \mathbb{R}^{N \times M \times 2L_j}$, which is the part of Hyper Synthesis output $\mathcal{H} \in \mathbb{R}^{N \times M \times 2C}$ corresponding to $\widehat{\mathcal{Y}}_j$. The concatenated tensor is input to EP_j, which consists of three convolutional layers with 3×3 kernels (Conv (3) in the figure), to estimate Gaussian means and variances for subsequent entropy coding. Quantized hyperpriors for all sub-latents (input to Hyper Synthesis) are encoded as side bitstream. We experimentally observed that the side bitstream accounts for 2–3% of total bits, on average.

At the decoder, hyperpriors are reconstructed from the side bitstream. The base-layer sub-latent $\widehat{\mathcal{Y}}_1$ is reconstructed from the the base bitstream using the hyperpriors. $\widehat{\mathcal{Y}}_1$ is sufficient for object detection. In case input reconstruction is also needed, $\widehat{\mathcal{Y}}_2$ is reconstructed from the enhancement bitstream using the hyperpriors, and the input image is reconstructed from $\widehat{\mathcal{Y}}_1 \cup \widehat{\mathcal{Y}}_2$.

Although this example involves a 2-task (2-layer) system for simplicity of explanation, it is possible to construct scalable systems with more layers by following the same principles. In fact, in Section IV, we also demonstrate a 3-task (3-layer) system that supports object detection (base layer), object segmentation (first enhancement layer) and input reconstruction (second enhancement layer). In this system, the latent space $\widehat{\mathcal{Y}}$ is partitioned into three parts: $\{\widehat{\mathcal{Y}}_1, \widehat{\mathcal{Y}}_2, \widehat{\mathcal{Y}}_3\}$, where $\widehat{\mathcal{Y}}_1$ supports object detection, $\widehat{\mathcal{Y}}_1 \cup \widehat{\mathcal{Y}}_2$ supports object segmentation, and the whole latent space $\widehat{\mathcal{Y}}_1 \cup \widehat{\mathcal{Y}}_2 \cup \widehat{\mathcal{Y}}_3$ supports input reconstruction.
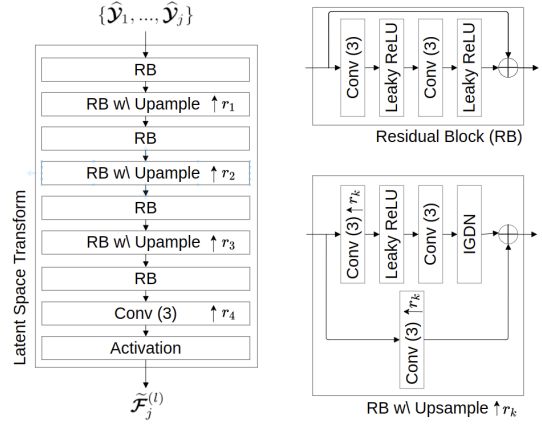
C. Task-dependent latent space transform

The base layer $\widehat{\mathcal{Y}}_1$ will support the base computer vision task, say object detection, for example. However, $\widehat{\mathcal{Y}}_1$, in general, does not match any of intermediate latent spaces of any of the object detectors. Hence, we introduce the latent space transform (LST) to map $\widehat{\mathcal{Y}}_1$ to an intermediate latent space of an object detector we want to use at the decoder. Specifically, let $\mathcal{F}_1^{(l)}$ be the feature tensor at the output of the l-th layer of a chosen object detector V_1, that is, $\mathcal{F}_1^{(l)} = V_1^{\text{(front-end)}}(\widehat{\mathbf{X}}, \psi)$, as in (2). Then the goal of the base LST is to map $\widehat{\mathcal{Y}}_1$ to $\mathcal{F}_1^{(l)}$. More generally, for the j-th task, we use $\{\widehat{\mathcal{Y}}_1, ..., \widehat{\mathcal{Y}}_j\}$ from the encoded latent space, and the goal of j-th LST is to map $\{\widehat{\mathcal{Y}}_1, ..., \widehat{\mathcal{Y}}_j\}$ to $\mathcal{F}_j^{(l)} = V_j^{\text{(front-end)}}(\widehat{\mathbf{X}}, \psi)$, where V_j is the DNN implementing the j-th task.

Fig. 6 shows the structure of our LST, which consists of residual blocks (RBs) similar to those in the Synthesis Decoder [16]. A RB consists of two convolutional layers with 3×3 kernels (Conv (3) in the figure) followed by Leaky ReLU. A RB with upsampling factor r_k (shown as $\uparrow r_k$) increases spatial dimension by r_k and also applies the inverse GDN [40]. Upsampling factors r_k are chosen based on the dimensions of $\{\widehat{\mathcal{Y}}_1, ..., \widehat{\mathcal{Y}}_j\}$ and $\mathcal{F}_j^{(l)}$. The last activation function is set to be the same as for the output of $V_j^{\text{(front-end)}}$, to produce a similar dynamic range for the output tensor. The output of the LST is $\widetilde{\mathcal{F}}_j^{(l)}$, an approximation to the desired $\mathcal{F}_j^{(l)}$, and the LST is trained to minimize their difference.

D. Information steering through learning

To steer the task-relevant information into the corresponding parts of the encoder latent space $\widehat{\mathcal{Y}}$, we construct a loss function in the form of a rate-distortion Lagrangian

$$\mathcal{L} = R + \lambda \cdot D, \qquad (4)$$

where R is the rate estimate, D is the combined distortion of various tasks, and λ is the Lagrange multiplier. As in [15],

$$R = \underbrace{\mathbb{E}_{x \sim p_x}[-\log_2 p_{\hat{y}}(\hat{y})]}_{\text{latent}} + \underbrace{\mathbb{E}_{x \sim p_x}[-\log_2 p_{\hat{z}}(\hat{z})]}_{\text{hyper-priors}}, \qquad (5)$$

where x denotes input data, $\hat{y}$ is the quantized latent data and $\hat{z}$ is the quantized hyper-prior. Distortion D is computed as

$$D = \text{MSE}(\mathbf{X}, \widehat{\mathbf{X}}) + \sum_{j=1}^{S-1} \gamma_j \cdot \text{MSE}\left(\boldsymbol{\mathcal{F}}_j^{(l)}, \widetilde{\boldsymbol{\mathcal{F}}}_j^{(l)}\right), \quad (6)$$

where γ_j are scale factors for various task distortions and S is the total number of tasks (i.e., layers). Besides training the model to accurately reconstruct task-relevant latent spaces, distortion terms $\text{MSE}\left(\boldsymbol{\mathcal{F}}_j^{(l)}, \widetilde{\boldsymbol{\mathcal{F}}}_j^{(l)}\right)$ also serve to steer the task-relevant information into the corresponding portions of the encoder latent space $\widehat{\boldsymbol{\mathcal{Y}}}$. This is because $\widetilde{\boldsymbol{\mathcal{F}}}_j^{(l)}$ only depends on $\{\widehat{\boldsymbol{\mathcal{Y}}}_1, ..., \widehat{\boldsymbol{\mathcal{Y}}}_j\}$ (through LST) and not the rest of the encoder latent space $\widehat{\boldsymbol{\mathcal{Y}}} \setminus \{\widehat{\boldsymbol{\mathcal{Y}}}_1, ..., \widehat{\boldsymbol{\mathcal{Y}}}_j\}$. Hence, information relevant to task j is backpropagated to $\{\widehat{\boldsymbol{\mathcal{Y}}}_1, ..., \widehat{\boldsymbol{\mathcal{Y}}}_j\}$. Further insight is provided through information flow measurements below.

E. Information flow

In this section we provide further insight into the operation of the proposed scalable framework from an information-theoretic perspective. Empirical information measurements are performed on a 2-layer system whose base task is object detection using a YOLOv3 [27] back-end, and the enhancement task is input reconstruction. The LST maps the base-layer features $\boldsymbol{\mathcal{Y}}_1$ to $\widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}$ of YOLOv3. The network was trained for 300 epochs with $\gamma_1 = 0.006$ in (6). Other training details are presented in Section IV-A.

Entropy estimates of the base features $\boldsymbol{\mathcal{Y}}_1$ and the entire latent representation $\boldsymbol{\mathcal{Y}}$ are obtained using rate estimates [15]

$$\begin{aligned} H(\boldsymbol{\mathcal{Y}}_1) &\approx \mathbb{E}_{x \sim p_x}\left[-\log_2 p_{\hat{y}_1}(\hat{y}_1)\right], \\ H(\boldsymbol{\mathcal{Y}}) &\approx \mathbb{E}_{x \sim p_x}\left[-\log_2 p_{\hat{y}}(\hat{y})\right]. \end{aligned} \quad (7)$$

Fig. 7 shows the results. The right-hand side Y-axis shows $H(\boldsymbol{\mathcal{Y}})$ in bits per pixel (bpp), computed as total bits divided by input image resolution. The left-hand side Y-axis shows the percentage of object detection information out of the total rate, that is

$$\frac{H(\boldsymbol{\mathcal{Y}}_1)}{H(\boldsymbol{\mathcal{Y}})} \cdot 100\%. \quad (8)$$

Fig. 7(a) shows the results for $\lambda = 0.0483$ in (4), while Fig. 7(b) shows the results for $\lambda = 0.0035$. In both cases, the number of base channels $i \in \{64, 96, 128\}$.

From Fig. 7, it can be seen that as λ decreases from 0.0483 in Fig. 7(a) to 0.0035 in Fig. 7(b), the total rate reduces from the range $[1.45, 1.55]$ bpp down to $[1.10, 1.17]$ bpp. This is clear from (4) because, as λ decreases, the importance of R in the Lagrangian cost increases, so there is more pressure to reduce the rate. The same happens with the object detection information rate. With $i = 128$ feature channels dedicated to object detection, in Fig. 7(a) the object detection information rate is approximately $0.8 \cdot 1.45 = 1.16$ bpp, and this falls down to approximately $0.92 \cdot 1.13 = 1.04$ bpp in Fig. 7(a), after 300 epochs. The same trend holds for other i's.

We can also notice that the fraction of the total rate devoted to the base layer does not scale proportionally to the fraction of the latent space occupied by the base layer. For example,
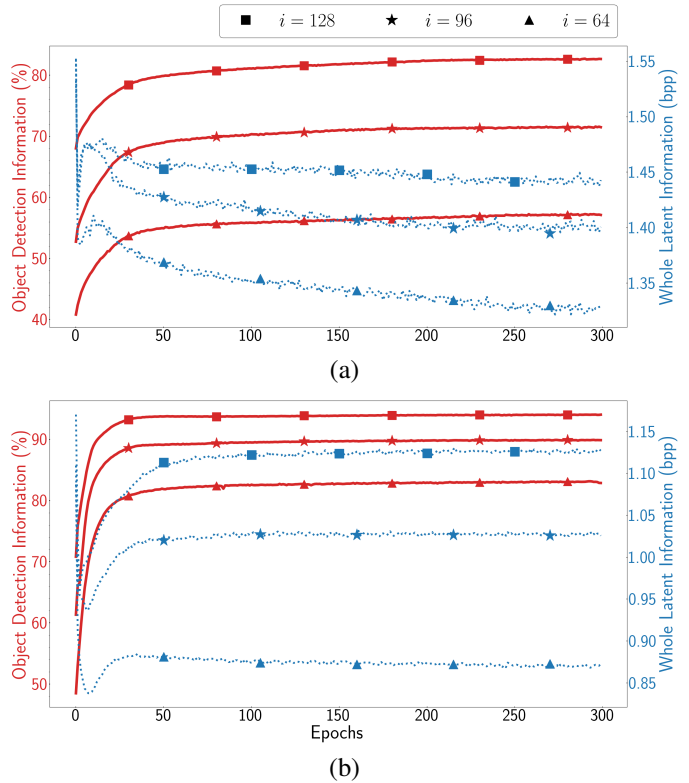

(a)

(b)

Fig. 7. Evolution of object detection information and total rate through training, when the number of latent-space channels dedicated to object detection is $i \in \{64, 96, 128\}$. Blue curves show the total rate $H(\boldsymbol{\mathcal{Y}})$, with the scale shown on the right vertical axis. Red curves show the percentage of object detection information (8) out of the total rate, with the scale shown on the left vertical axis. Sub-figure (a) corresponds to the model trained with $\lambda = 0.0483$ in (4), and (b) to the model trained with $\lambda = 0.0035$ in (4).

when the number of base layer channels halves from $i = 128$ to $i = 64$, the percentage of the total rate it occupies only reduces from 80% to 55% in Fig. 7(a), and from from 92% to 82% in Fig. 7(b). This is due to the fact that the base layer serves two purposes - it is responsible for object detection, and also plays a role in input reconstruction. Hence, through RD optimization, it receives more rate than would be expected from the fraction of the latent space it occupies.

It is well known that scalable extensions of conventional codecs, such as H.264/AVC [41] and H.265/HEVC [42], suffer a rate increase of about 15-25% per enhancement layer compared to single-layer coding [43] in terms of the rate-distortion performance. This is due to information redundancy between layers. It is natural to ask whether such redundancy exists in our scalable coding framework? Ideally, $I(\widehat{\boldsymbol{\mathcal{Y}}}_1; \widehat{\boldsymbol{\mathcal{Y}}}_2) = 0$, meaning that base and enhancement layer are independent, and they can be coded separately without loss of coding efficiency. If this were the case, then because $\widehat{\boldsymbol{\mathcal{Y}}}_1 \rightarrow \widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}$ through LST, we would have $I(\widehat{\boldsymbol{\mathcal{Y}}}_2; \widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}) = 0$. To test this, we measure the mutual information between $\widehat{\boldsymbol{\mathcal{Y}}}_j, j \in \{1, 2\}$ and $\widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}$. Mutual information measurements are obtained using the Kernel Density Estimator (KDE) from [44], as follows.

We first divide sub-latent tensors $\widehat{\boldsymbol{\mathcal{Y}}}_j$ into fibers $\widehat{\boldsymbol{\mathcal{Y}}}_j \in \mathbb{R}^{1 \times 1 \times L_j}$, where $L_1 = 128$ and $L_2 = 64$. Also $\widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}$ is
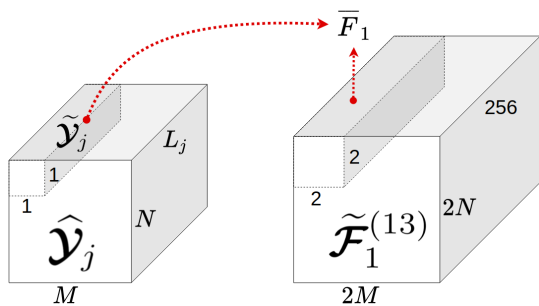
Fig. 8. Mutual information estimation between $\widehat{\boldsymbol{\mathcal{Y}}}_j$ and $\widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}$

Fig. 9. Estimated mutual information (MI) $I(\widetilde{\boldsymbol{\mathcal{Y}}}_1; \overline{F}_1)$ and $I(\widetilde{\boldsymbol{\mathcal{Y}}}_2; \overline{F}_1)$

divided into spatially corresponding fibers with the size of $2 \times 2 \times 256$, as illustrated in Fig. 8. Then, we cluster the estimated output feature fibers into $K = 16$ clusters (similarly to [45]) with cluster index denoted $\overline{F}_1$, and then compute

$$
\begin{aligned}
I(\widetilde{\boldsymbol{\mathcal{Y}}}_j; \overline{F}_1) &= H(\widetilde{\boldsymbol{\mathcal{Y}}}_j) - H(\widetilde{\boldsymbol{\mathcal{Y}}}_j | \overline{F}_1) \\
&= H(\widetilde{\boldsymbol{\mathcal{Y}}}_j) - \sum_{\bar{f}_1 \in \{1, \dots, K\}} p(\bar{f}_1) \cdot H(\widetilde{\boldsymbol{\mathcal{Y}}}_j | \overline{F}_1 = \bar{f}_1).
\end{aligned}
\tag{9}
$$

Due to clustering, $I(\widetilde{\boldsymbol{\mathcal{Y}}}_j; \overline{F}_1)$ is an underestimate of $I(\widetilde{\boldsymbol{\mathcal{Y}}}_j; \widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)})$, but the measurements still provide useful insight into the behavior of the system. Fig. 9 shows the evolution of the estimated mutual information over 400 epochs for $\lambda = 0.0483$ and $i = 128$. Since $\widehat{\boldsymbol{\mathcal{Y}}}_1 \rightarrow \widetilde{\boldsymbol{\mathcal{F}}}_1^{(13)}$, we see that $I(\widetilde{\boldsymbol{\mathcal{Y}}}_1; \overline{F}_1)$ remains relatively high, around 3.76 bits per fiber, throughout the 400 epochs. Meanwhile, $I(\widetilde{\boldsymbol{\mathcal{Y}}}_2; \overline{F}_1)$ reduces down to 1.20 bits per fibre, showing that the enhancement layer carries less and less information about object detection as the training goes on. While $I(\widetilde{\boldsymbol{\mathcal{Y}}}_2; \overline{F}_1)$ does not reach the ideal value of zero, the decreasing trend is evident, which confirms that the loss function (4)-(6) provides appropriate information steering to various portions of the latent space. At the same time, since the base and enhancement latents are coded independently, a non-zero mutual information between them is an indication of inefficiency of scalable coding. Indeed, as will be seen in Section IV-C, in terms of input reconstruction, our 2-layer system is less efficient than [16], while our 3-layer system is less efficient than our 2-layer system, even though all three systems share the same encoder architecture. This loss of efficiency due to scalability has also been observed in earlier studies on scalable coding [43], so it is not a new phenomenon. We believe our approach of looking at the problem through the lens of mutual information will contribute to further advances in this area.

F. Extensions to new tasks

One may also ask what happens if a new task needs to be added to the system post-hoc, after the initial system has already been trained? This is a challenge for all coding-for-machine approaches, because the features trained for initial tasks may not be appropriate for the new task. In general, some modifications need to be made, but there are several possibilities, depending on how efficient the new system needs
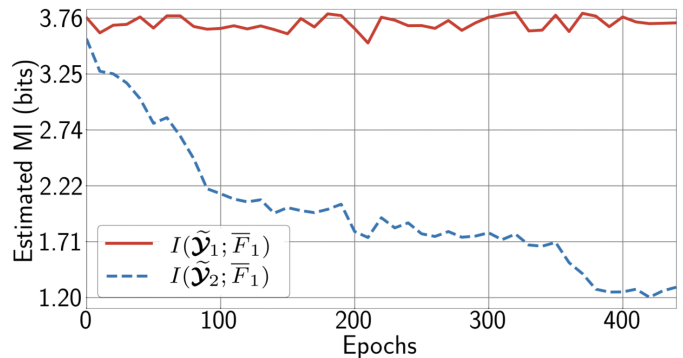
to be and whether or not it is possible to retrain the encoder. Four our proposed systems, the possibilities are as follows:

1) If the new task happens to be a subset of an existing task,[2] then one can simply train a new LST from the portion of the latent space dedicated to the previous task into the latent space of the new back-end, because all the information is already available there. In this case, encoder does not need to be retrained, only the LST.
2) If the new task is not a subset of any of the previous tasks, one could retrain the encoder to include the new task and allocate a portion of the latent space to it.
3) As shown in Section III-A, any task T is a subset of the input reconstruction task. Therefore, one can simply train a LST from the full latent space into the latent space of the new back-end. This would avoid retraining the encoder, but the new task would likely operate at a higher bitrate compared to approach 2) above, because the full latent space needs to be decoded.

IV. EXPERIMENTS

Here we describe the experiments to assess the performance of 2- and 3-layer networks presented earlier. Our choice of machine vision tasks (object detection, object segmentation) and the corresponding back-end models (YOLOv3 [27] and FPN [38]) is motivated by the MPEG VCM activities [2], where these are some of the recommended tasks and models. Individual tasks of each network are evaluated on the relevant datasets and the results are compared with the relevant benchmarks. Both 2- and 3-layer networks are trained on the same datasets using a two-stage training strategy described below.

A. Training setup

Our multi-task networks are trained in two stages. In the first stage, the networks are trained on CLIC [20] and JPEG-AI [47] datasets. Randomly cropped patches with size of 256×256 from both datasets are used as input. We also set the mini-batch size to 16. Adam optimizer with a fixed learning rate of 10^{-4} is used for 400 epochs on a GeForce RTX 2080 GPU with 11 GB RAM. Then, we change the dataset

---

[2]For example, an existing task is detection of 10 different breads of dogs and 5 breeds of cats, while the new task is just detecting 'dogs' and 'cats'.
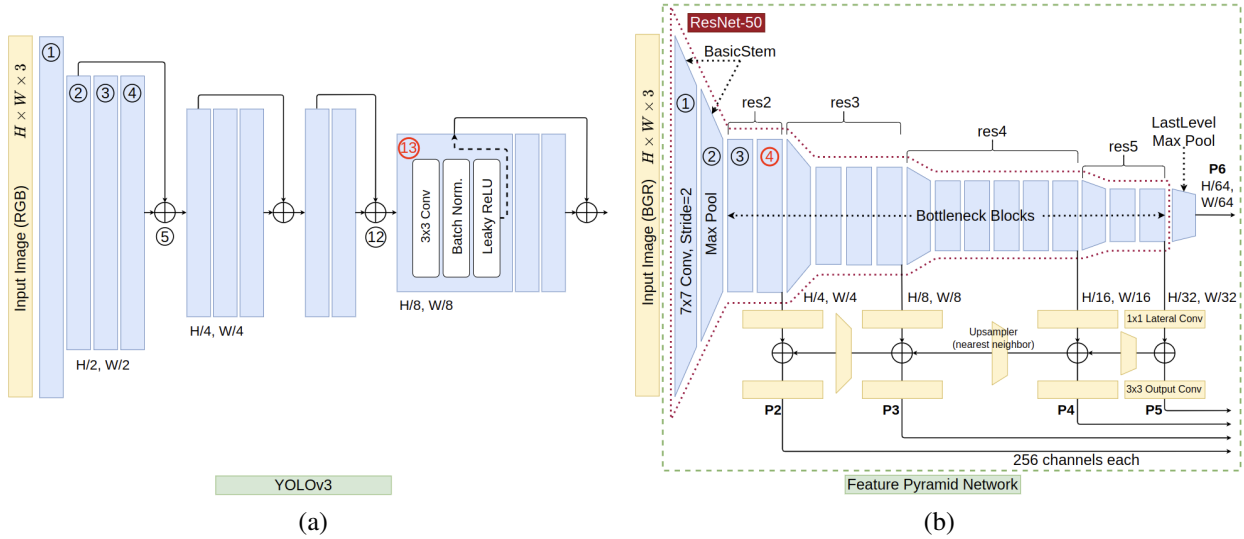
Fig. 10. Part of network architectures for targeted vision tasks: (a) YOLOv3 [27] and (b) Feature Pyramid Network used for Faster [46] and Mask [28] R-CNN

TABLE I
λ VALUES FOR TRAINING MODELS

Quality Index	1	2	3	4	5	6
λ	0.0018	0.0035	0.0067	0.013	0.025	0.0483

to VIMEO-90K [48] to continue the training for another 300-400 epochs in the second stage. Likewise, randomly cropped patches are drawn from the dataset, but this time the learning rate decreases with polynomial decay for every 10 epochs. Six values of λ, shown in Table I, are used in (4) to produce six versions of the trained networks, as in [49]. The total number of latent-space channels is set to $C = 192$ in each case, as in [14]–[16]. Specific details of the 2- and 3-layer networks are given below.

Two-layer network: The 192 latent-space channels are split into $L_1 = 128$ base-layer channels and $L_2 = 64$ enhancement layer channels. A LST in the base layer maps the base features $\widehat{\mathcal{Y}}_1 \in \mathbb{R}^{N \times M \times 128}$ to the target features at the convolution output of layer 13 of YOLOv3, $\widetilde{\mathcal{F}}_1^{(13)} \in \mathbb{R}^{2N \times 2M \times 256}$. To match the resolution of the target feature tensor, the LST scaling factors r_k, $k \in \{1, 2, 3, 4\}$, are set to 2, 1, 1, and 1, respectively. The processing at layer 13 of YOLOv3 includes a convolutional layer followed by batch normalization and Leaky ReLU activation, as shown in Fig 10(a). Since the pre-trained weights of YOLOv3 [27] represent prior knowledge obtained over the training data, we keep and re-use them for the batch normalization followed by an activation function at layer 13, so the last layer of the LST simply adopts a linear activation. The estimate of the layer 13 convolution output, produced by our LST, is then used as input to the batch normalization with the learned weights at layer 13. To account for this, the distortion equation is slightly modified from (6), to

$$D = \text{MSE}(\mathbf{X}, \widehat{\mathbf{X}}) + \gamma \cdot \text{MSE}\left(\mathcal{F}_1^{(13)}, V^{(13)}(\widetilde{\mathcal{F}}_1^{(13)}, \rho^*)\right), \tag{10}$$

where $\gamma = 0.006$ and $V^{(13)}$ includes batch normalization followed by Leaky ReLU activation, with pre-trained weights ρ^* from [27].

Three-layer network: Here, the 192 latent-space channels are split into $L_1 = 96$ channels for the base layer (which will support object detection), $L_2 = 32$ channels for the first enhancement layer (which will support segmentation) and the remaining $L_3 = 64$ channels for the last enhancement layer, which will support input reconstruction. LSTs in the base and the first enhancement layer individually estimate intermediate tensors of Faster R-CNN [46] for object detection and Mask R-CNN [28] for segmentation, respectively. Fig. 10(b) presents the ResNet-50 [23] based Feature Pyramid Network (FPN) [38] used as a backbone network for both R-CNN networks. In particular, the LSTs estimate the outputs of layer 4 in the FPN, $\widetilde{\mathcal{F}}_j^{(4)} \in \mathbb{R}^{4N \times 4M \times 256}$, where $j \in \{1, 2\}$. Hence, to generate the correct size of the feature tensors from the sub-latents $\widetilde{\mathcal{Y}}_1$ and $\{\widehat{\mathcal{Y}}_1, \widehat{\mathcal{Y}}_2\}$, both LSTs have the same configuration of scaling factors: $r_1 = r_2 = 2$ and $r_3 = r_4 = 1$. The activation at layer 4 in the FPN is ReLU, so we use the same function for the last activation layer for both LSTs. For distortion computation, the MSE is measured at various points $\mathbf{P}2$–$\mathbf{P}6$ in the FPN, which are shown in Fig. 10(b). To account for this, the distortion equation (6) is slightly modified to

$$D = \text{MSE}(\mathbf{X}, \widehat{\mathbf{X}}) + \gamma \cdot \frac{1}{5} \cdot \sum_{j=1}^{2} \sum_{l=2}^{6} \text{MSE}(\mathbf{P}l_j, V_{\text{FPN,j}}^{\text{back-end},\mathbf{P}l}(\widetilde{\mathcal{F}}_j^{(4)}, \rho_j^*)) \tag{11}$$

where $\gamma = 0.0015$ and $V_{\text{FPN,j}}^{\text{back-end},\mathbf{P}l}$ represents the portion of the FPN back-end up to $\mathbf{P}l$, with pre-trained weights ρ_j^* [50], using $\widetilde{\mathcal{F}}_j^{(4)}$ as input.

B. Evaluation on machine vision tasks

Our multi-task networks are evaluated on relevant datasets associated with targeted tasks, in terms of task accuracy vs. bitrate.
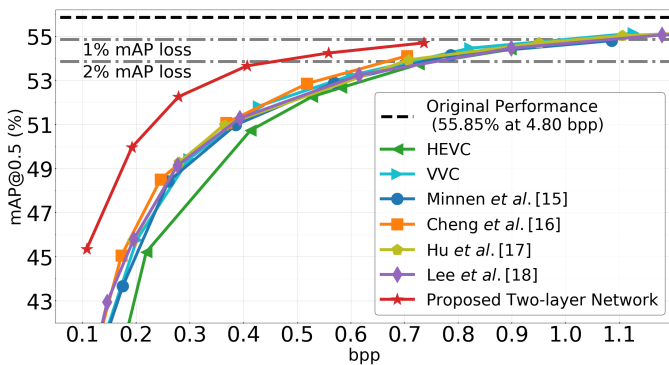
Fig. 11. Object detection performance of the two-layer network compared with benchmarks on the COCO2014 validation set

TABLE II
SUMMARIZED PERFORMANCE OF VISION TASKS AGAINST VARIOUS BENCHMARKS WITH BD METRICS

Proposed network	Two-layer		Three-layer			
	Object Detection				Segmentation	
Benchmark	BD-rate	BD-mAP	BD-rate	BD-mAP	BD-rate	BD-mAP
VVC	-39.8	2.79	-70.5	2.33	-71.2	2.34
HEVC	-47.9	4.55	-73.2	3.05	-74.7	2.96
[15]	-41.3	3.26	-78.7	3.73	-77.2	3.38
[16]	-37.4	2.89	-76.6	3.62	-75.4	3.49
[17]	-38.1	1.99	-76.3	2.80	-75.6	2.56
[18]	-39.6	2.93	-77.5	3.66	-76.2	3.42

Benchmarks: We compare our proposed networks against several benchmarks, including the latest standard codecs such as HEVC [42] and VVC [51], as well as five recent DNN-based image codecs [15]–[18]. Specifically, the reference software versions for HEVC and VVC are HM-16.20 [52] and VTM-12.3 [53], and are used in all intra configuration [54], [55] with quantization parameters QP $\in \{22, 25, 28, ..., 40\}$. To encode RGB images using the standard codecs, these images are first converted to YUV444 in accordance with ITU-R BT.709 [56], then the chrominance channels are sub-sampled to create the YUV420 version of the image, which can then be fed to an HEVC or VVC encoder. After decoding, chrominance channels of the YUV420 output are interpolated to YUV444 using bilinear interpolation, then converted to RGB in accordance with ITU-R BT.709 [56]. DNN-based codecs are trained to compress RGB images directly, so no conversion to YUV is needed. Finally, decoded RGB images are used as input to the pre-trained computer vision models[3] to examine task-specific accuracy.

Two-layer network: Our two-layer network supports object detection in the base layer using the YOLOv3 [27] back-end. We first evaluate the object detection performance on the COCO2014 validation set [57], which includes about 5,000 images. Since most vision networks resize the input to a specific resolution before processing, we also resize input images to 512×512 using bilinear interpolation without letterboxing, in order to generate (via LST) a feature tensor $\widetilde{\mathcal{F}}_1^{(13)} \in \mathbb{R}^{64 \times 64 \times 256}$ that can be directly fed into the YOLOv3 back-end. The same resizing is done with benchmark image codecs.

Fig. 11 presents a comparison of our object detection performance against various benchmarks in terms of bitrate in bits per pixel (bpp) vs. mean Average Precision (mAP), where bpp is computed by dividing the total number of coded bits (base and side bitstreams) by the number of input pixels. The mAP uses the Intersection of Union (IoU) threshold of 0.5. In the figure, the black dashed line shows the default mAP performance of 55.85% when using the COCO2014 validation images as input to YOLOv3 [27] with pre-trained weights. The average bitrate of the images in the COCO2014 validation

---

[3]In each case, the same model whose back-end is used in our multi-task network.

---

set, coded with JPEG, is 4.80 bpp, which is indicated in the legend in the figure. Our object detection achieves the best rate-accuracy performance, with mAP loss of about 1% at 0.74 bpp, where most benchmarks suffer a mAP loss of about 2%. Moreover, at 0.56 bpp, our method operates within a 2% mAP loss margin, whereas most benchmarks have lost about 4% mAP at this point. Cheng *et al.* [16] shows the best performance among the benchmarks, but there is still significant gap between it and our proposed method.

Table II (first three columns) summarizes object detection vs. bitrate results using extended versions of Bjøntegaard Delta (BD) metric [58], [59]. For the BD-mAP metric, positive numbers represent an average increase of mAP at the same bitrate. For BD-rate, negative numbers indicate average bit savings at the same accuracy. Our method shows a noticeable bit savings and increased accuracy compared to all benchmarks. For example, against HEVC, we achieve BD-rate savings of –47.9%, and BD-mAP gain of 4.55%. Against Cheng *et al.* [16], we achieve BD-rate savings of –37.4%, and BD-mAP gain of 2.89%. The greatest bit reduction of 41.3% is accomplished against Minnen *et al.* [15], also marginally less significant reductions are achieved against other DNN-based networks [17], [18].

Three-layer network: The three-layer network supports object detection in the base layer and object segmentation in the first enhancement layer. The corresponding back-ends use ResNet-50-based Faster [46] and Mask [28] R-CNN with pre-trained weights [50], respectively. Since the benchmarks' performance of two tasks using those R-CNNs are evaluated on the COCO2017 validation set [57] and publicly reported via Detectron2 [50], we use the same dataset for fair evaluation and comparison with our proposed network. Faster R-CNN and Mask R-CNN have a constraint on the input resolution that the shorter edge must be less than or equal to 800 pixels according to given default configuration. Hence, we resize the test images to meet the constraint using bilinear interpolation prior to the experiment, then use the resized images as input to our three-layer network. As a result, generated feature tensors $\widetilde{\mathcal{F}}_1^{(4)}$ and $\widetilde{\mathcal{F}}_2^{(4)}$ from the base and the first enhancement layer can be fed into the Faster and Mask R-CNN back-ends, respectively. The resized images are also used as input to benchmark image codecs.

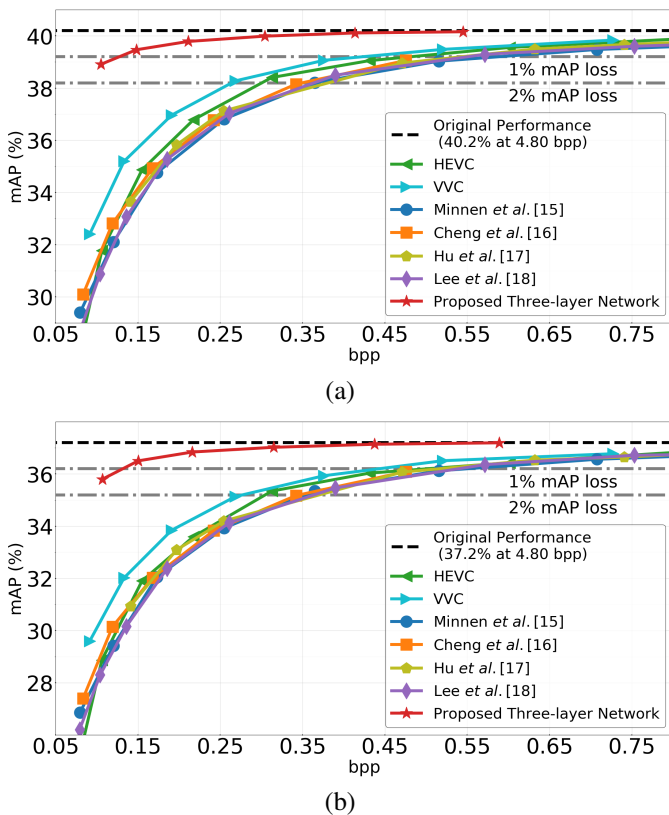Fig. 12 shows the performance of our three layer network and the benchmarks on both tasks. In Fig. 12(a) and (b),

(a)

(b)

Fig. 12. Performance of (a) object detection and (b) segmentation of the three-layer network compared with benchmarks on the COCO2017 validation set

TABLE III
BD-RATE RELATIVE TO VARIOUS BENCHMARKS ON KODAK [60]

Benchmarks	Proposed methods			
	Two-layer Network		Three-layer Network	
	BD-rate (PSNR)	BD-rate (MS-SSIM)	BD-rate (PSNR)	BD-rate (MS-SSIM)
VVC	10.17%	-7.83%	30.43%	2.14%
HEVC	-14.27%	-26.15%	1.38%	-17.96%
JPEG	-64.52%	-64.06%	-57.28%	-57.84%
[15]	-3.58%	-7.83%	14.02%	2.06%
[16]	4.49%	-1.90%	24.24%	9.55%
[17]	-5.58%	-16.84%	12.19%	-4.92%
[18]	-7.01%	-15.23%	9.94%	-7.37%
Two-layer Network	-		18.84%	11.95%

Benchmarks: Our network performance is compared with the benchmarks, which include standard codecs [42], [51] and DNN-based codecs [15]–[18], all optimized for MSE. The results reported here for HEVC and VVC are borrowed from CompressAI [49]. For HEVC [42] and VVC [51], HM-16.20 [52] with range extension profile and VTM-9.1 [61] were used, respectively. RGB images were first converted to YUV444 in accordance with BT.709 [56], then directly input to the codecs. Decoded images in the YUV444 format were converted back to RGB in accordance with BT.709 [56]. Since each input image is coded separately, all intra configurations following the common test conditions [54], [55] was used. Images were coded using various QPs from 12 to 47 with a step size of 5, and we show the RD points that cover the range of bitrates acheived by DNN-based codecs. We also borrowed JPEG results within this bitrate range from CompressAI [49].

Fig. 13(a) shows PSNR (RGB) vs. bitrate curves. VVC achieves the best performance among all methods in this figure. The method from which our backbone is derived, Cheng *et al.* [16], achieves the second-best performance. Our 2-layer network shows competitive performance compared to [16] at lower bitrates, but the reconstruction quality slightly degrades (by about 0.3dB) compared to [16] at higher bitrates. This is the price to pay for scalability and supporting object detection in the base layer. Compared to [17], [18] and other conventional codecs, our network still outperforms, while simultaneously supporting scalability. Table III shows BD-rate [58], [59] comparisons among various codecs. Overall, our 2-layer network has a loss of 10.17% against VVC and 4.49% against [16], but saves –3.58%, –5.58%, –7.01%, – 14.27%, and –63.99% of bits compared to Minnen *et al.* [15], Hu *et al.* [17], Lee *et al.* [18], HEVC, and JPEG, respectively, while providing scalability.

Our 3-layer network is less efficient in terms of rate-distortion performance on input reconstruction. As shown in the last row of Table III, the 3-layer network suffers an increase of 18.84% BD-rate compared to the 2-layer network. Nonetheless, it is worthwhile to remark that earlier work on conventional scalable codecs suggests that adding one

dashed black lines show the default mAP performance of 40.2% and 37.2% at 4.80 bpp on average[4] for object detection and segmentation, respectively. Here, mAP is obtained using the IoU threshold from 0.5 to 0.95 with a step size of 0.05. On both tasks, our 3-layer network shows excellent performance: less than 1% mAP drop down to 0.15 bpp, and less than about 1.5% mAP drop at 0.1 bpp. Meanwhile, all benchmarks lose 1% mAP already at 0.4 bpp, while at 0.1 bpp, they have lost 7-8% mAP.

Table II (last four columns) shows object detection and segmentation performance vs. bitrate in terms of extended versions of BD metrics [59]. Here, the gains of our network against the benchmarks are even higher. Against VVC, which was the best-performing benchmark, our network achieves BD-rate savings of –73.2% on object detection and –71.2% on object segmentation. At the same time, the BD-mAP gains against VVC are 2.33% on object detection and 2.34% on object segmentation.

C. Evaluation on input reconstruction

The highest enhancement layer of our 2- and 3-layer networks supports input reconstruction for human viewing. The performance here is examined on the Kodak dataset [60], which includes 24 uncompressed RGB images.

[4]Although the COCO2014 and COCO2017 validation sets are not identical, the average bitrate of images in each set, coded by JPEG, is 4.80 bpp.
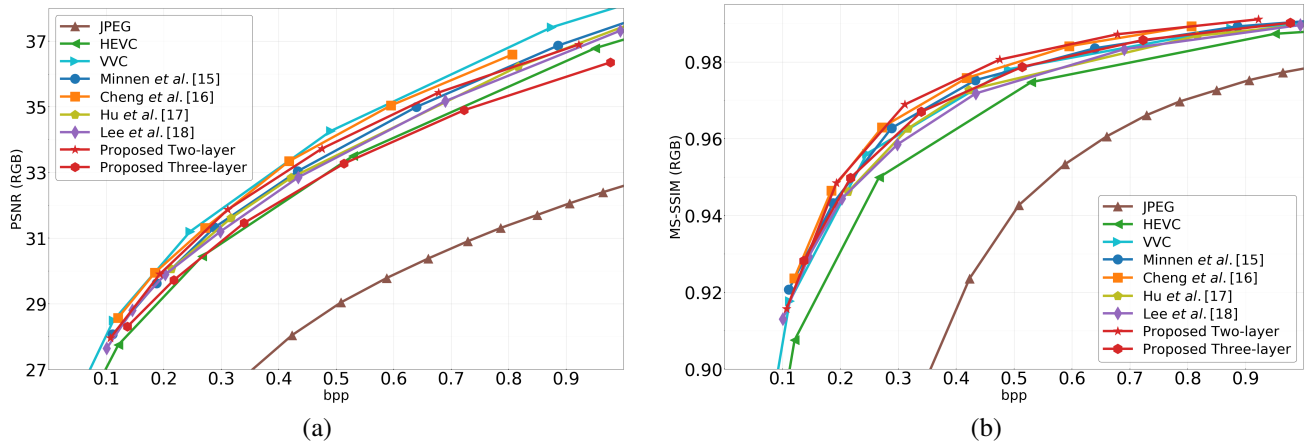
Fig. 13. Comparison of input reconstruction performance on Kodak in terms of (a) PSNR vs. bpp and (b) MS-SSIM vs. bpp

scalability layer costs about 15-25% in terms of BD-rate [43], so the performance of our 3-layer network seems reasonable in this context. It is still comparable with HEVC (1.38% increase in BD-rate) and much better than JPEG, while providing 3-layer scalability and superior efficiency on computer vision tasks. Overall, Fig. 13(a) shows that both our 2- and 3-layer networks are comparable with state of the art codecs in terms of PSNR at lower bitrates, but their relative PSNR performance degrades at higher bitrates.

In addition to PSNR results, we also provide performance comparisons in terms of MS-SSIM [62] vs. bitrate in Fig. 13(b). It is known that DNN-based codecs perform very well on MS-SSIM (even if they are optimized for MSE), and this is also evident in Fig. 13(b) and Table III. In particular, our 2-layer network achieves the best results in this comparison, showing coding gains against all benchmarks, with –1.9% savings compared to the best benchmark [16]. Even our 3-layer network now shows a gain of almost –18% relative to HEVC, and comparable performance (with a gap around 2%) relative to VVC and [15]. We believe this improved performance of our networks in terms of MS-SSIM is due to better latent representations related to objects features associated with machine vision tasks in the lower layers, which encourages higher structural quality of the reconstructed input.

Finally, we consider a comparison with [35]. While direct comparison is not possible since the tasks (and the corresponding image datasets) are different, an indirect comparison via BPG (HEVC Intra) is possible based on presented results. According to Fig. 9 in [35], when it comes to input reconstruction, the approach in [35] is less efficient than HEVC in terms of PSNR, and comparable to HEVC on MS-SSIM. On the other hand, from Table III, our 2-layer network is 14.27% more efficient than HEVC on PSNR, while our 3-layer network is slightly (about 1.3%) less efficient than HEVC in terms of PSNR. Meanwhile, both our networks are significantly more efficient than HEVC Intra on MS-SSIM. Hence, it appears that our networks are more efficient than [35] in terms of input reconstruction.

TABLE IV
AVERAGE EXECUTION TIME (IN SECONDS) FOR DNN-BASED MODELS

Avg. Time (s)	DNN-based benchmarks				Proposed methods	
	[15]	[16]	[17]	[18]	Two-layer	Three-layer
Encode	4.12	4.05	5.60	20.52	9.36	12.96
Decode	8.29	8.30	6.25	70.88	17.26	24.94

D. Run-time complexity of DNN-based methods

We compare the run time complexity of our methods with DNN-based benchmarks [15]–[18] by measuring the average encoding and decoding time per image on a GeForce RTX 2080 GPU with 11 GiB RAM. We evaluate the DNN-based codecs on the Kodak dataset [60]. Our methods target the input reconstruction at the highest enhancement layer, but entail extra computation related to sub-latent coding in the lower layer(s) for machine vision task(s). Meanwhile, the benchmarks target only input reconstruction. Three benchmarks [15]–[17] and ours are implemented based on CompressAI [49], while [18] is developed using Tensorflow-Compression [63]. All models run on a single GPU, but some of the computation for some models [16], [18], including ours, has to run on a CPU (e.g., auto-regressive context modelling for entropy coding), which slows the system down.

Table IV presents run-time complexity of the models, and ours has the second-largest encoding and decoding time, after [18]. Compared to [16], which is the baseline network we used as the backbone for our models, run-time complexity increases in accordance (roughly proportionally) with the number of layers. This seems reasonable. Nonetheless, complexity is an issue for all DNN-based codecs, including ours.

E. Ablation study for LST

In the proposed approach to latent space scalability, the LST plays a crucial role of transforming a portion of the encoder's latent space into another latent space containing task-relevant features. In this section, we perform an ablation study on the LST in our 2-layer network, by gradually removing residual
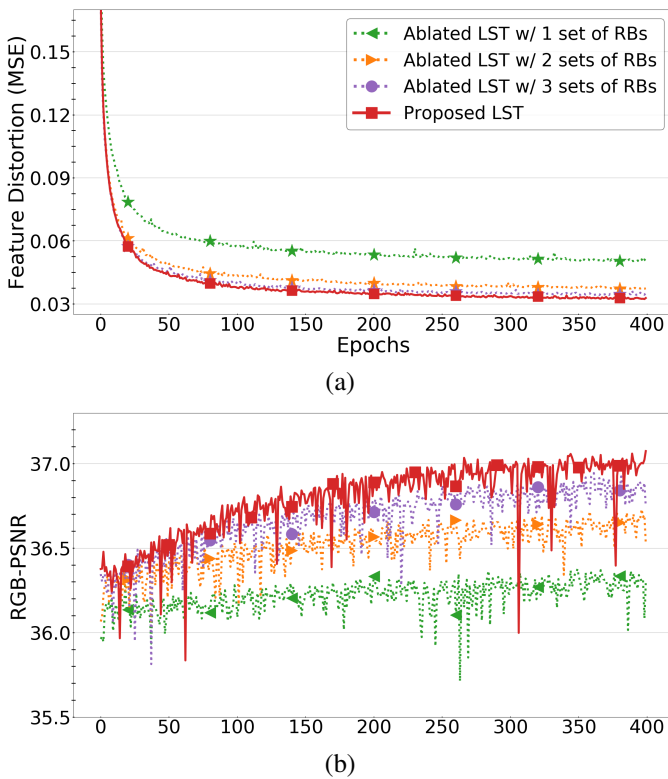
(a)

(b)

Fig. 14. LST ablation results on the 2-layer network: (a) feature distortion in the base layer, (b) input reconstruction in the enhancement layer.
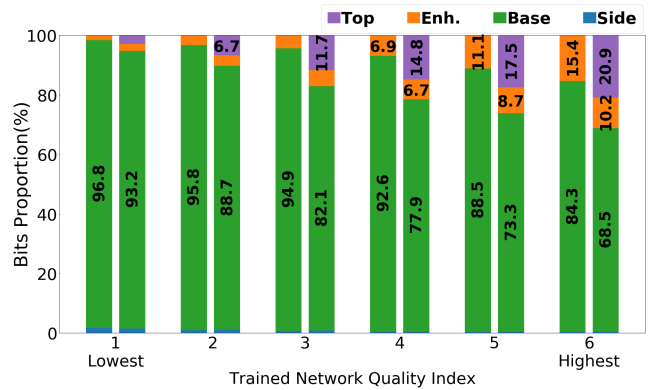

Fig. 15. Proportion of bits in various layers of the bitstream. There are six quality indices, corresponding to six values of λ shown in Table I. For each quality index, the left bar corresponds to the 2-layer network, and the right bar to the 3-layer network.

F. Bitstream analysis

Fig. 15 shows the bitsteam composition of six versions of our networks, from those trained for the lowest bitrate (quality index 1), to those trained for the highest bitrate (quality index 6). For each index, left bar corresponds to the 2-layer network, and the right bar to the 3-layer network. There are relatively few bits in the side bitstream (blue), which accounts for less than 2% of the total bitstream. The base layer (green) accounts for most bits, and its fraction drops from over 93% at lowest bitrates to less than 85% at highest bitrates for the 2-layer network, and less than 70% for the 3-layer network. The first enhancement layer (orange, denoted "Enh." in the figure) and the second enhancement layer in the 3-layer network (purple, denoted "Top") take up progressively more bits at higher bitrates. Considering the fact that our networks show more competitive performance of input reconstruction at lower bitrates (Fig. 13), these results seem to indicate that the base layer conveys significant information for input reconstruction as well, in addition to enabling object detection.

G. Visual examples

One of the key contributions of the present paper is that our scalable DNN-based image coding approach is able to support birate-efficient high-quality machine vision without resorting to input reconstruction. Fig. 16 shows two examples of the outputs of our 3-layer network, along with the results obtained by the benchmarks. For each example, the first row shows the input image and the reconstructed images with the corresponding bitrate and RGB PSNR (bpp/dB). The next two rows show the results of object segmentation and object detection. For the benchmarks, reconstructed image is fed to the corresponding model (Faster R-CNN [46] for detection, Mask R-CNN [28] for segmentation) with pre-trained weights to obtain the results. For our 3-layer network, only the corresponding part of the bitstream is decoded. Hence, since the input is not reconstructed in these cases, the results are shown on the empty background, and the corresponding rate is indicated below the image.

In the first example, our network successfully detects and segments all three objects, with bitrates of 0.195 bpp for

blocks (RBs, shown in Fig. 6) and examining the impact of such removal on both outputs of the network. The study is performed on the highest-rate 2-layer model ($\lambda = 0.0483$) for the first 400 epochs of training. The results are shown in Fig. 14. Fig. 14(a) shows the feature distortion in the target latent space (which is the latent space of YOLOv3) for four cases: LST with one set of RBs,[5] LST with two sets of RBs, LST with three sets of RBS, and the proposed LS, which has four sets of RBs. As seen in the figure, increasing the number of RBs improves the ability of the LST to match the target features of YOLOv3 and reduces the distortion in the target latent space. With four sets of RBs (the proposed LST), we seem to have reached diminishing returns in terms of performance vs. complexity, so four sets of RBs seems to be a reasonable choice for the LST.

Fig. 14(b) shows the impact of LST ablation on input reconstruction. Even though the LST is in the object detection processing pipeline, since the network is trained end-to-end, a change in the LST has an impact on input reconstruction as well. We see that the performance of input reconstruction improves as the number of sets of RBs increases to four, which is the proposed LST. In essence, the ability of LST to better extract object detection-relevant features from the base layer helps the system better distribute the information between the base and enhancement layers, thereby improving both tasks.

---

[5]One "set of RBs" consists of one RB and one RB w/ Upsample, illustrated in Fig. 6.

detection and 0.205 bpp for segmentation. In contrast, all benchmarks lead to mislabelling of a horse on the right as a person, even in the case of object detection, despite the fact they use more bits than our base layer. In the second example, benchmark-coded images lead to some missing objects, while our network correctly detect them all. For example, the image coded by [15] leads to missing the second person from the right in the background, as well as the baseball. Also, image coded by [16] leads to missing the person in the background. With conventional codecs (HEVC and VVC), either the person or the baseball are missed. These examples illustrate why our 3-layer network provides superior performance in terms of object detection and segmentation in Fig. 12.

V. CONCLUSION

We presented a DNN-based image compression framework with latent-space scalability for human and machine vision. Latent image representation is coded into multiple layers, which can be separately decoded to enable the required task. We embodied the proposed ideas into 2- and 3-layer multi-task networks supporting object detection, segmentation, and input reconstruction. Mutual information estimates show that the proposed loss function facilitates steering of relevant task-specific information into the corresponding portions of the latent space during training. The experiments show that our multi-task networks provide 37% - 80% bitrate savings on machine vision tasks compared to relevant benchmarks, while being comparable to state of the art image codecs in terms of input reconstruction quality.

REFERENCES

[1] L. Duan, J. Liu, W. Yang, T. Huang, and W. Gao, "Video coding for machines: A paradigm of collaborative compression and intelligent analytics," *IEEE Trans. Image Process.*, vol. 29, pp. 8680–8695, 2020.

[2] "Call for evidence for video coding for machines," ISO/IEC JTC 1/SC 29/WG 2, m55065, Oct. 2020.

[3] J. Ascenso, "JPEG AI use cases and requirements," ISO/IEC JTC 1/SC29/WG1 M90014, Jan. 2021.

[4] B. Shen and I. K. Sethi, "Direct feature extraction from compressed images," in *Proc. SPIE Storage and Retrieval for Image and Video Databases IV*, 1996, vol. 2670.

[5] Z. Qian, W. Wang, and T. Qiao, "An edge detection method in DCT domain," in *Int. Workshop Inform. Electron. Eng.*, 2012, pp. 344–348.

[6] J. Nightingale, Q. Wang, C. Grecos, and S. R. Goma, "Deriving video content type from HEVC bitstream semantics," in *Proc. SPIE Real-Time Image and Video Processing*, 2014, vol. 9139, pp. 1–13.

[7] S. H. Khatoonabadi and I. V. Bajić, "Video object tracking in the compressed domain using spatio-temporal Markov random fields," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 300–313, 2012.

[8] H. Choi and I. V. Bajić, "Corner proposals from HEVC bitstreams," in *Proc. IEEE ISCAS*, 2017, pp. 1–4.

[9] H. Choi and I. V. Bajić, "HEVC intra features for human detection," in *Proc. IEEE GlobalSIP*, 2017, pp. 393–397.

[10] S. R. Alvar, H. Choi, and I. V. Bajić, "Can you tell a face from a HEVC bitstream?," in *Proc. IEEE MIPR*, 2018, pp. 257–261.

[11] S. R. Alvar, H. Choi, and I. V. Bajić, "Can you find a face in a HEVC bitstream?," in *Proc. IEEE ICASSP*, 2018, pp. 1288–1292.

[12] ISO/IEC JTC 1, "Compact descriptors for visual search," 2015, ISO/IEC 15938-13.

[13] D. Lowe, "Distinctive image feature from scale-invariant keypoints," *Int. J. Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[14] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. ICLR*, 2018.

[15] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *Advances in Neural Information Processing Systems*, vol. 31, pp. 10771–10780, 2018.

[16] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proc. IEEE/CVF CVPR*, 2020.

[17] Y. Hu, W. Yang, and J. Liu, "Coarse-to-fine hyper-prior modeling for learned image compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 11013–11020.

[18] J. Lee, S. Cho, and S.-K. Beack, "Context-adaptive entropy model for end-to-end optimized image compression," *arXiv preprint arXiv:1809.10452*, 2018.

[19] Y. Xie, K. Cheng, and Q. Chen, "Enhanced invertible encoding for learned image compression," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 162–170.

[20] "Challenge on learned image compression (CLIC)," [Online]: http://www.compression.cc/.

[21] D. Ding, Z. Ma, D. Chen, Q. Chen, Z. Liu, and F. Zhu, "Advances in video compression system using deep nueral network:A review and case studies," *arXiv preprint arXiv:2101.06341*, Jan. 2021.

[22] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF CVPR*, 2016, pp. 770–778.

[24] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. IEEE/CVF CVPR*, 2017, pp. 3156–3164.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. ECCV*, 2016, pp. 21–37.

[27] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, Apr. 2018.

[28] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE/CVF ICCV*, 2017, pp. 2961–2969.

[29] R. Torfason, F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Towards image understanding from deep compression without decoding," in *Proc. ICLR*, 2018.

[30] H. Choi and I. V. Bajić, "Near-lossless deep feature compression for collaborative intelligence," in *Proc. IEEE MMSP*, Aug. 2018.

[31] S. R. Alvar and I. V. Bajić, "Multi-task learning with compressible features for collaborative intelligence," in *Proc. IEEE ICIP'19*, Sep. 2019, pp. 1705–1709.

[32] S. R. Alvar and I. V. Bajić, "Bit allocation for multi-task collaborative intelligence," in *Proc. IEEE ICASSP*, 2020, pp. 4342–4346.

[33] S. R. Alvar and I. V. Bajić, "Pareto-optimal bit allocation for collaborative intelligence," *IEEE Trans. Image Process.*, vol. 30, pp. 3348–3361, 2021.

[34] Y. Hu, S. Yang, W. Yang, L.-Y. Duan, and J. Liu, "Towards coding for human and machine vision: A scalable image coding approach," in *Proc. IEEE ICME*, 2020, pp. 1–6.

[35] K. Liu, D. Liu, L. Li, N. Yan, and H. Li, "Semantics-to-signal scalable image compression with learned revertible representations," *Int. J. Comput. Vis.*, pp. 1–17, Jun. 2021.

[36] H. Choi and I. V. Bajić, "Latent-space scalability for multi-task collaborative intelligence," in *IEEE ICIP*, Sep. 2021, arXiv:2105.10089.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[38] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE/CVF CVPR*, 2017, pp. 936–944.

[39] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 2nd edition, 2006.

[40] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," *arXiv preprint arXiv:1511.06281*, 2015.

[41] Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int/Electrotech. Commun. (ISO/IEC JTC 1), "Advanced video coding," Rec. ITU-T H.264 and ISO/IEC 14496-10, 2005.

[42] Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int/Electrotech. Commun. (ISO/IEC JTC 1), "High efficiency video coding," Rec. ITU-T H.265 and ISO/IEC 23008-2, 2019.

[43] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian, "Overview of SHVC: scalable extensions of the high efficiency video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 20–34, 2016.

Fig. 16. Illustration of object detection, segmentation, and input reconstruction results for several benchmark methods and our 3-layer network on two images: the top three rows correspond to the first image, and the bottom three rows to the second image. For each input image, the top row shows input reconstruction, the next row shows object segmentation, and the last row shows object detection. The first column shows the original images and the results of object detection and segmentation produced by the FPN-based R-CNNs applied to those images. The detection and segmentation results are overlaid on the original images. The next four columns show the images encoded and decoded by various benchmarks, and the FPN-based R-CNN results produced from decoded images, again overlaid on those images. Two numbers are shown beneath each image, indicating the bitrate and RGB PSNR, respectively, in the format bpp/PSNR(dB). The last column shows the outputs of our 3-layer network. Object detection and segmentation results are shown on a blank background, because our system does not need to reconstruct the input image in order to perform object detection or segmentation. For these cases, only the bitrate is shown beneath each result. To reduce the clutter, in all cases we only show objects with a confidence score of over 80%.

[44] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *Proc. ICLR*, 2018.

[45] S. Lee and I. V. Bajić, "Information flow through U-Nets," in *Proc. IEEE ISBI*, 2021, pp. 812–816.

[46] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2016.

[47] "JPEG AI dataset," [Online]: https://jpeg.org/jpegai/dataset.html.

[48] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vision*, vol. 127, no. 8, pp. 1106–1125, 2019.

[49] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "CompressAI: a PyTorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, 2020.

[50] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2,"

https://github.com/facebookresearch/detectron2, 2019.

[51] Int. Telecommun. Union-Telecommun. (ITU-T) and Int. Standards Org./Int/Electrotech. Commun. (ISO/IEC JTC 1), "Versatile video coding," Rec. ITU-T H.266 and ISO/IEC 23090-3, 2020.

[52] "HEVC reference software (HM 16.20)," [Online]: http://hevc.hhi. fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.20+SCM-8.8, Accessed: 2019-12-12.

[53] "VVC reference software (VTM 12.3)," [Online]: https://vcgit.hhi. fraunhofer.de/jvet/VVCSoftware_VTM/-/tags/VTM-12.3, Accessed: 2021-12-10.

[54] F. Bossen, "Common HM test conditions and software reference configurations," in ISO/IEC JTC1/SC29/WG11, JCTVC-L1100, Jan. 2013.

[55] J. Boyce, K. Suehring, X. Li, and V. Seregin, "JVET common test conditions and software reference configurations," in ISO/IEC JTC1/SC29/WG11, JVET-J1010, Apr. 2018.

[56] Int. Telecommun. Union-Radiocommunication. (ITU-R), "Parameter values for the hdtv standards for production and international programme exchange BT series," Rec. ITU-R BT.709-6, 2015.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in Proc. ECCV, Sept. 2014.

[58] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," https://www.itu.int/wftp3/av-arch/video-site/0104_Aus/VCEG-M33.doc, Apr. 2001, VCEG-M33.

[59] C. Hollmann, S. Liu, W. Gao, and X. Xu, "[VCM] On VCM reporting template," ISO/IEC JTC 1/SC 29/WG 2, m56185, Jan. 2021.

[60] E. Kodak, "Kodak lossless true color image suite (PhotoCD PCD0992)," http://r0k.us/graphics/kodak.

[61] "VVC reference software (VTM 9.1)," [Online]: https://vcgit.hhi. fraunhofer.de/jvet/VVCSoftware_VTM/-/tags/VTM-9.1, Accessed: 2021-12-10.

[62] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in Proc. IEEE Asilomar Conf. Signals, Systems & Computers, 2003, vol. 2, pp. 1398–1402.

[63] J. Ballé, S. J. Hwang, N. Johnston, D. Minnen, and E. Agustsson, "Tensorflow-Compression," [Online]: https://github.com/tensorflow/compression.

[64] I. V. Bajić, W. Lin, and Y. Tian, "Collaborative intelligence: Challenges and opportunities," in Proc. IEEE ICASSP, 2021, pp. 8493–8497.

[65] R. W. Yeung, A First Course in Information Theory, Kluwer Academic Publishers, 2002.

[66] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in Proc. IEEE Information Theory Workshop (ITW), 2015, pp. 1–5.

[67] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," arXiv preprint arXiv: 1703.00810v3, 2017.

Appendix A
Deep feature compressibility

In this appendix, we prove that intermediate latent representations in a non-generative DNN are at least as compressible as it's input, both in terms of lossless compression and lossy compression. This provides some theoretical support for DNN-based image codecs, as well as collaborative intelligence applications [64], where intermediate DNN features computed from sensed signals are sent from the edge to the cloud for inference. The source of randomness in our analysis is the selection of DNN input data, so unless otherwise stated, all expectations are over the input data distribution.

The proofs presented here are consequences of the Data Processing Inequality (DPI) [39], [65], which we review briefly here. Random variables X_1, X_2, ..., X_n form a Markov chain, indicated as $X_1 \rightarrow X_2 \rightarrow ... \rightarrow X_n$, if their joint distribution can be factored as $p(x_1, x_2, ..., x_n) = p(x_1)p(x_2|x_1)...p(x_n|x_{n-1})$. In that case, the chain also satisfies the Markov property in reverse, that is, $X_n \rightarrow ... \rightarrow X_2 \rightarrow X_1$ is also a Markov chain [65]. Note that $X \rightarrow X \rightarrow Y$ is trivially a Markov chain. It has been noted that successive

layers in non-generative feedforward networks form a Markov chain [66]. It is easy to see that if $f(\cdot)$ is a function, then $X \rightarrow Y \rightarrow f(Y)$ is a Markov chain. A crucial result related to Markov chains that we will make repeated use of is the data processing inequality (DPI) [65], which says that if $U \rightarrow X \rightarrow Y \rightarrow V$ is a Markov chain, then $I(U; V) \leq I(X; Y)$. A special case of DPI [39] states that for $X \rightarrow Y \rightarrow V$, $I(X; V) \leq I(X; Y)$.

For the DNN-related notation, input is denoted $\mathbf{X}$, l-th layer's output $\boldsymbol{\mathcal{Y}}^{(l)}$, and the DNN inference output is T. We will focus on non-generative feedforward networks, which do not introduce any randomness in their processing. Examples of such networks include most well-known DNNs, e.g., ResNet [23], YOLO [27], etc. In such networks, for fixed network weights, the output of any layer is uniquely determined by the input through feedforward processing.

First we focus on the lossless compression case. Consider a deep feedforward non-generative network and let $\{\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)}\}$ be the outputs of n of it's layers.

Theorem 1. *The joint entropy of $\{\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)}\}$ is upper bounded by the entropy of the input $\mathbf{X}$, that is,*

$$H(\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)}) \leq H(\mathbf{X}), \tag{12}$$

with equality if and only if the input $\mathbf{X}$ can be reconstructed exactly from $\{\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)}\}$.

Proof. Let $\boldsymbol{\mathcal{Y}} = (\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)})$. Note that $\boldsymbol{\mathcal{Y}}$ is a deterministic function of the input $\mathbf{X}$: since the model is non-generative, when $\mathbf{X}$ is input to the model, $\boldsymbol{\mathcal{Y}}$ is uniquely determined by passing the signal forward through to all the n layers of interest. Hence, when $\mathbf{X}$ is given, there is no uncertainty about $\boldsymbol{\mathcal{Y}}$, so $H(\boldsymbol{\mathcal{Y}}|\mathbf{X}) = 0$. By considering the Markov chain $\mathbf{X} \rightarrow \mathbf{X} \rightarrow \boldsymbol{\mathcal{Y}}$ and using the DPI, we can write

$$\begin{aligned} H(\mathbf{X}) = I(\mathbf{X}; \mathbf{X}) &\geq I(\mathbf{X}; \boldsymbol{\mathcal{Y}}) \\ &= H(\boldsymbol{\mathcal{Y}}) - H(\boldsymbol{\mathcal{Y}}|\mathbf{X}) \\ &= H(\boldsymbol{\mathcal{Y}}) \\ &= H(\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, \ldots, \boldsymbol{\mathcal{Y}}^{(n)}). \end{aligned} \tag{13}$$

This proves the claimed inequality. To see when equality holds in (13), we start with $H(\boldsymbol{\mathcal{Y}}) = I(\mathbf{X}; \boldsymbol{\mathcal{Y}})$, which is clear from (13), and expand mutual information as follows:

$$H(\boldsymbol{\mathcal{Y}}) = I(\mathbf{X}; \boldsymbol{\mathcal{Y}}) = H(\mathbf{X}) - H(\mathbf{X}|\boldsymbol{\mathcal{Y}}). \tag{14}$$

So, in order to have $H(\mathbf{X}) = H(\boldsymbol{\mathcal{Y}})$, we must have $H(\mathbf{X}|\boldsymbol{\mathcal{Y}}) = 0$. That is to say, the input $\mathbf{X}$ must be a deterministic function of $\boldsymbol{\mathcal{Y}}$, which means that one must be able to reconstruct the input $\mathbf{X}$ from the layers' outputs $\{\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)}\}$ exactly. □

Special cases of (13) have been presented in the literature on information bottleneck in [44], [67], but for a single layer rather than multiple intermediate layers. However, considering a collection of layers $\{\boldsymbol{\mathcal{Y}}^{(1)}, \boldsymbol{\mathcal{Y}}^{(2)}, ..., \boldsymbol{\mathcal{Y}}^{(n)}\}$ rather than just a single layer $\boldsymbol{\mathcal{Y}}^{(l)}$ is a practical necessity, because many useful models have skip/parallel connections across layers, so compression of multiple intermediate layers may be needed. The above theorem shows that the joint entropy of the outputs

of any collection of DNN layers is no larger than the entropy of the input. According to this result, in the limit, we should be able to compress (losslessly, for now) outputs from any number of internal DNN layers at least as efficiently as the input, so long as we compress them jointly.

We next turn our attention to lossy compression, which is somewhat more involved. The central concept in lossy compression the *rate-distortion function* $R(D)$, which, for a source $\mathbf{X}$, is defined as [39]:

$$R(D) = \min_{p(\hat{\mathbf{x}}|\mathbf{x}) \ : \ \sum_{(\mathbf{x},\hat{\mathbf{x}})} [p(\mathbf{x})p(\hat{\mathbf{x}}|\mathbf{x})d(\mathbf{x},\hat{\mathbf{x}})] \ \leq \ D} I(\mathbf{X}; \widehat{\mathbf{X}}). \quad (15)$$

In the above equation, $\widehat{\mathbf{X}}$ is the quantized version of $\mathbf{X}$ – this process of quantization, where values of $\mathbf{X}$ are represented by another, generally smaller set of values $\widehat{\mathbf{X}}$, is what causes "loss" in lossy compression. For a given pair $(\mathbf{x}, \hat{\mathbf{x}})$, the discrepancy is measured by a distortion metric $d(\mathbf{x}, \hat{\mathbf{x}})$. The summation below the $\min$ operator computes the expected distortion $\mathbb{E}[d(\mathbf{X}, \widehat{\mathbf{X}})]$ with respect to the joint distribution $p(\mathbf{x}, \hat{\mathbf{x}})$. This expected distortion is required to be no larger than some value D. Within the joint distribution $p(\mathbf{x}, \hat{\mathbf{x}})$, the distribution of $\mathbf{X}$, $p(\mathbf{x})$, is fixed, and the minimization is carried out over the conditional (quantizer) distribution $p(\hat{\mathbf{x}}|\mathbf{x})$. The quantizer can be deterministic, in which case $p(\hat{\mathbf{x}}|\mathbf{x})$ is a delta function for any given $\mathbf{x}$, or random, in which case $p(\hat{\mathbf{x}}|\mathbf{x})$ is a proper, non-degenerate distribution. The theory is general enough to accommodate both cases. The source coding theorem and its converse [39] show that $R(D)$, as defined above, is the minimum achievable rate[6] per source symbol that results in expected distortion of at most D. Hence, $R(D)$ represents a fundamental bound on lossy compression, just like entropy represents a fundamental bound on lossless compression.

In order to use rate-distortion theory to analyze lossy feature compression, we will need several additional concepts. First, let the function implemented by the DNN from input $\mathbf{X}$ to output T be $f(\cdot)$, so that the DNN's output is $T = f(\mathbf{X})$. Let the mapping from the input $\mathbf{X}$ to the set of n layer outputs $\mathcal{Y} = (\mathcal{Y}^{(1)}, \mathcal{Y}^{(2)}, ..., \mathcal{Y}^{(n)})$, so that $\mathcal{Y} = g(\mathbf{X})$, and the mapping from $\mathcal{Y}$ to the output T be $h(\cdot)$ so that $T = h(\mathcal{Y})$. Note that $T = h(g(\mathbf{X}))$, so $f = h \circ g$ is the composition of g and h. These relationships are summarized below:

$$\mathbf{X} \xrightarrow{g} \mathcal{Y} \xrightarrow{h} T \qquad (16)$$

with f spanning from $\mathbf{X}$ to T.

We will measure distortion at the output of the model. This is one difference with respect to the conventional rate-distortion formulation, where distortion $d(\mathbf{x}, \hat{\mathbf{x}})$ measures how much the quantized $\hat{\mathbf{x}}$ deviates from $\mathbf{x}$. In our case, what is relevant is how the output of the model is affected by quantization, so our distortion is formulated using the mapping to the output, as $d(f(\mathbf{x}), f(\hat{\mathbf{x}}))$. The choice of the distortion metric itself depends on the model; for regression models, squared error distortion is the natural choice, while for classification models, cross-entropy seems appropriate. The key result

---

below, however, is agnostic to the choice of the distortion metric. Using the distortion measured at the output, we can define the set of all conditional distributions (quantizers) $p(\hat{\mathbf{x}}|\mathbf{x})$ that achieve output distortion of no more than D:

$$\mathcal{P}_{\mathbf{X}}(D) = \left\{ p(\hat{\mathbf{x}}|\mathbf{x}) \ : \ \mathbb{E}\left[d\left(f(\mathbf{X}), f(\widehat{\mathbf{X}}) \right) \right] \leq D \right\}. \quad (17)$$

Using this set, we can formulate the rate-distortion function for the model's input $\mathbf{X}$ as:

$$R_{\mathbf{X}}(D) = \min_{p(\hat{\mathbf{x}}|\mathbf{x}) \ \in \ \mathcal{P}_{\mathbf{X}}(D)} I(\mathbf{X}; \widehat{\mathbf{X}}). \quad (18)$$

Since we also need to consider quantization of intermediate layers $\mathcal{Y}$, we analogously define the set of all conditional distributions (quantizers) $p(\hat{\mathbf{y}}|\mathbf{y})$ that achieve output distortion of no more than D:

$$\mathcal{P}_{\mathcal{Y}}(D) = \left\{ p(\hat{\mathbf{y}}|\mathbf{y}) \ : \ \mathbb{E}\left[d\left(h(\mathcal{Y}), h(\widehat{\mathcal{Y}}) \right) \right] \leq D \right\}. \quad (19)$$

Note that in (19), we use $h(\cdot)$, the mapping from $\mathcal{Y}$ to T, in the expected distortion. Using this set, we formulate the rate-distortion function for the model's intermediate layers $\mathcal{Y}$:

$$R_{\mathcal{Y}}(D) = \min_{p(\hat{\mathbf{y}}|\mathbf{y}) \ \in \ \mathcal{P}_{\mathcal{Y}}(D)} I(\mathcal{Y}; \widehat{\mathcal{Y}}). \quad (20)$$

We now state and prove the lossy compression result.

Theorem 2. *For any distortion level $D \geq 0$, the minimum achievable rate for compressing $\mathcal{Y}$ is upper bounded by the minimum achievable rate for compressing $\mathbf{X}$, that is,*

$$R_{\mathcal{Y}}(D) \leq R_{\mathbf{X}}(D). \quad (21)$$

The necessary condition for equality is that $\mathbf{X}$ can be reconstructed exactly from $\mathcal{Y}$.

Proof. Let $D \geq 0$ be given, and let $p^*(\hat{\mathbf{x}}|\mathbf{x}) \in \mathcal{P}_{\mathbf{X}}(D)$ be the conditional distribution that achieves the rate-distortion bound for $\mathbf{X}$, i.e., solves (18). This conditional distribution defines a quantized representation $\widehat{\mathbf{X}}$ of the model's input $\mathbf{X}$.

Now let us draw inputs $\mathbf{X}$ according to the data distribution $p(\mathbf{x})$, quantize $\mathbf{X}$ into $\widehat{\mathbf{X}}$ using $p^*(\hat{\mathbf{x}}|\mathbf{x})$ and then pass the quantized input $\widehat{\mathbf{X}}$ through $g(\cdot)$, which is the part of the model that computes the intermediate layers of interest. This will generate $\widetilde{\mathcal{Y}}$, which is the intermediate layers' output when the input is $\widehat{\mathbf{X}}$, as shown below.

$$\mathbf{X} \xrightarrow[p^*(\hat{\mathbf{x}}|\mathbf{x})]{} \widehat{\mathbf{X}} \xrightarrow{g} \widetilde{\mathcal{Y}} \qquad (22)$$

In parallel, consider the deterministic mapping from $\mathbf{X}$ to $\mathcal{Y}$ defined by the model:

$$\mathbf{X} \xrightarrow{g} \mathcal{Y} \qquad (23)$$

Together, the last two equations mean that for any *particular* input $\mathbf{X}$, we can obtain a *particular* $\mathcal{Y}$ according to (23) and a (possibly degenerate) *distribution* of $\widetilde{\mathcal{Y}}$ according to (22). Pairing up the values of $\mathcal{Y}$ with the corresponding distributions of $\widetilde{\mathcal{Y}}$ implicitly defines a conditional distribution of $\widetilde{\mathcal{Y}}$ given $\mathcal{Y}$, which we will denote as $q(\tilde{\mathbf{y}}|\mathbf{y})$. In addition, let $q(\mathbf{y})$ be the distribution of $\mathcal{Y}$ induced by $p(\mathbf{x})$ through $\mathcal{Y} = g(\mathbf{X})$.

First, we show that $q(\tilde{\mathbf{y}}|\mathbf{y}) \ \in \ \mathcal{P}_{\mathcal{Y}}(D)$, that is to say, $q(\tilde{\mathbf{y}}|\mathbf{y})$ satisfies the distortion constraint for quantizing $\mathcal{Y}$.

To see this, note that $p^*(\hat{\mathbf{x}}|\mathbf{x}) \in \mathcal{P}_{\mathbf{X}}(D)$, which means that using $p^*(\hat{\mathbf{x}}|\mathbf{x})$, we have $\mathbb{E}[d(f(\mathbf{X}), f(\widehat{\mathbf{X}}))] \leq D$. But because $f(\mathbf{X}) = h(g(\mathbf{X})) = h(\mathcal{Y})$ and $f(\widehat{\mathbf{X}}) = h(g(\widehat{\mathbf{X}})) = h(\widetilde{\mathcal{Y}})$, we have

$$
\begin{aligned}
\mathbb{E}\left[d\left(h(\mathcal{Y}), h(\widetilde{\mathcal{Y}})\right)\right] &= \sum_{(\mathbf{y},\tilde{\mathbf{y}})} [q(\mathbf{y}) \cdot q(\tilde{\mathbf{y}}|\mathbf{y}) \cdot d(h(\mathbf{y}), h(\tilde{\mathbf{y}}))] \\
&= \sum_{(\mathbf{x},\hat{\mathbf{x}})} [p(\mathbf{x}) \cdot p^*(\hat{\mathbf{x}}|\mathbf{x}) \cdot d(f(\mathbf{x}), f(\hat{\mathbf{x}}))] \\
&= \mathbb{E}\left[d\left(f(\mathbf{X}), f(\widehat{\mathbf{X}})\right)\right] \\
&\leq D,
\end{aligned}
\tag{24}
$$

where the second equality follows from the fact that $q(\mathbf{y})$ and $q(\tilde{\mathbf{y}}|\mathbf{y})$ are induced by $p(\mathbf{x})$ and $p^*(\hat{\mathbf{x}}|\mathbf{x})$, respectively, through $g(\cdot)$. Therefore, $q(\tilde{\mathbf{y}}|\mathbf{y})$ satisfies the distortion constraint for quantizing $\mathcal{Y}$, so $q(\tilde{\mathbf{y}}|\mathbf{y}) \in \mathcal{P}_{\mathcal{Y}}(D)$.

Next, we show that $I(\mathcal{Y}; \widetilde{\mathcal{Y}}) \leq I(\mathbf{X}; \widehat{\mathbf{X}})$. Note that $\mathbf{X} \to \widehat{\mathbf{X}} \to \widetilde{\mathcal{Y}}$ is a Markov chain based on (22), and therefore the reverse chain $\widetilde{\mathcal{Y}} \to \widehat{\mathbf{X}} \to \mathbf{X}$ must also be a Markov chain. We now add one more processing step to the end of this chain, $\mathcal{Y} = g(\mathbf{X})$, to obtain the following Markov chain: $\widetilde{\mathcal{Y}} \to \widehat{\mathbf{X}} \to \mathbf{X} \to \mathcal{Y}$. Applying the DPI to this chain, we conclude $I(\mathcal{Y}; \widetilde{\mathcal{Y}}) \leq I(\mathbf{X}; \widehat{\mathbf{X}})$. This is true for any $p(\hat{\mathbf{x}}|\mathbf{x})$. But for the optimal $p^*(\hat{\mathbf{x}}|\mathbf{x})$, which achieves the rate-distortion bound for $\mathbf{X}$, and its induced distribution $q(\tilde{\mathbf{y}}|\mathbf{y})$, we have

$$
I(\mathcal{Y}; \widetilde{\mathcal{Y}}) \leq I(\mathbf{X}; \widehat{\mathbf{X}}) = R_{\mathbf{X}}(D). \tag{25}
$$

Together, (24) and (25) show that $q(\tilde{\mathbf{y}}|\mathbf{y})$ is one distribution in $\mathcal{P}_{\mathcal{Y}}(D)$ that achieves mutual information $I(\mathcal{Y}; \widetilde{\mathcal{Y}}) \leq R_{\mathbf{X}}(D)$. Therefore, $R_{\mathcal{Y}}(D)$, which is the minimum $I(\mathcal{Y}; \widehat{\mathcal{Y}})$ over all distributions $p(\hat{\mathbf{y}}|\mathbf{y}) \in \mathcal{P}_{\mathcal{Y}}(D)$, cannot be larger than $R_{\mathbf{X}}(D)$:

$$
R_{\mathcal{Y}}(D) = \min_{p(\hat{\mathbf{y}}|\mathbf{y}) \in \mathcal{P}_{\mathcal{Y}}(D)} I(\mathcal{Y}; \widehat{\mathcal{Y}}) \leq R_{\mathbf{X}}(D). \tag{26}
$$

We have therefore established the inequality in (21). We now turn to conditions that are needed for equality in (21). The necessary condition for equality to hold in (21) is that equality holds in the DPI (25). For this to happen, we must be able to recover $\widehat{\mathbf{X}}$ from $\widetilde{\mathcal{Y}}$, or in general, recover $\mathbf{X}$ from $\mathcal{Y}$. In other words, this is a necessary condition for $R_{\mathcal{Y}}(D) = R_{\mathbf{X}}(D)$. But it is not a sufficient condition, because even if we had $I(\mathcal{Y}; \widetilde{\mathcal{Y}}) = R_{\mathbf{X}}(D)$ in (25), this only holds for $q(\tilde{\mathbf{y}}|\mathbf{y})$, which is induced by $p^*(\hat{\mathbf{x}}|\mathbf{x})$; the minimization in (26) over all conditional distributions in $\mathcal{P}_{\mathcal{Y}}(D)$ may still produce a lower $I(\mathcal{Y}; \widehat{\mathcal{Y}})$. This completes the proof of the theorem. $\square$

The above theorem is a lossy counterpart to Theorem 1, showing that in the lossy case also, latent representations are at least as compressible as the input for any given distortion level. What is interesting in this case is that even if the mapping from input $\mathbf{X}$ to latent representation $\mathcal{Y}$ is perfectly invertible, $\mathcal{Y}$ may still be more compressible than $\mathbf{X}$, because it may allow for more efficient quantization. This is different from the lossless case where invertibility of the mapping from $\mathbf{X}$ to $\mathcal{Y}$ meant than $\mathcal{Y}$ is no more compressible than $\mathbf{X}$.