

# LATENT-SPACE SCALABILITY FOR MULTI-TASK COLLABORATIVE INTELLIGENCE

*Hyomin Choi and Ivan V. Bajić*

School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada

## ABSTRACT

We investigate latent-space scalability for multi-task collaborative intelligence, where one of the tasks is object detection and the other is input reconstruction. In our proposed approach, part of the latent space can be selectively decoded to support object detection while the remainder can be decoded when input reconstruction is needed. Such an approach allows reduced computational resources when only object detection is required, and this can be achieved without reconstructing input pixels. By varying the scaling factors of various terms in the training loss function, the system can be trained to achieve various trade-offs between object detection accuracy and input reconstruction quality. Experiments are conducted to demonstrate the adjustable system performance on the two tasks compared to the relevant benchmarks.

**Index Terms**— Deep feature compression, collaborative intelligence, multi-task models, latent-space scalability, video coding for machines.

## 1. INTRODUCTION

Rapid deployment of artificial intelligence (AI)-enabled applications is putting a strain on computational resources across a number of systems, from handheld devices to large-scale cloud computing systems. Recent studies [1, 2] have established the concept of *collaborative intelligence* (CI) as one way to address such challenges, by splitting an AI model (e.g., a deep neural network, DNN) between the edge and the cloud. In such a framework, intermediate features, produced by the model’s front-end, are sent from the edge to the cloud. Hence, compression of intermediate features has become a topic of interest. Related standardization activities include Video Coding for Machines (VCM) [3] and JPEG-AI [4].

For example, [5–9] have demonstrated that coding intermediate features can lead to significant compression gains, with a minimal loss in task accuracy. These studies were based on off-the-shelf single-task DNN models. In our earlier work [10], a multi-task CI model was developed that supports object detection and input reconstruction, using near-lossless coding of intermediate features. Related approaches [11, 12], utilizing lossy feature compression, were presented for different multi-task models. Unlike these methods, where a single feature tensor is coded to support multiple back-end tasks, re-

cent proposals [13, 14] focus on scalable coding to support multiple tasks. For example, [14] presented a scalable coding approach that supports facial landmark detection and generative input face reconstruction. While the generative decoder works well for face reconstruction, it might be less successful in reconstructing non-face details of the input image.

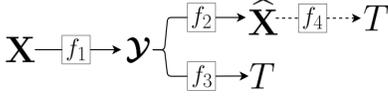
In this paper, we present a CI system that uses *latent-space scalability* to support object detection and input image reconstruction. Specifically, a part of the latent space (base layer) is dedicated to object detection (base task) while the entire latent space is utilized for input reconstruction. The portion of the latent space that is not used for the base task can be interpreted as an enhancement layer. Such representation can also be used for other multi-task models (i.e., the base task could be something other than object detection) and allows for efficient, scalable learnt representation of the input.

Section 2 briefly reviews related approaches to intermediate feature compression. The proposed method is described in Section 3. Experimental results are presented in Section 4, followed by conclusions in Section 5.

## 2. RELATED WORK

Early approaches to feature compression [5–9] focused on coding a single feature tensor from a single-task DNN, with tasks being image classification [6, 7] or object detection [5]. A popular approach for coding feature tensors in these works was to tile the tensor into an image, apply pre-quantization (say, to 8 bits per tensor element), and then use a conventional image codec for compression. In order to further improve the tensor coding efficiency, [8, 9] proposed additional methods such as tensor channel prediction and data clipping.

Since multiple tasks are often required in image/video analysis [13, 15], another group of methods has focused on feature compression for multi-task DNNs [10–12]. Although these works validated the idea that multi-task analytics are possible from a single compressed feature tensor, no further study was made as to how to efficiently organize the latent space for multiple tasks. In particular, in these approaches, reconstruction of the whole tensor is needed to accomplish any task. Most recently, [14] proposed a scalable feature representation for coding face images. Specifically, edge maps needed for facial landmark detection form the base layer, while additional color information forms the enhancement



**Fig. 1.** Markov chain model for our CI system.

layer. Facial landmark detection can be accomplished from the base-layer information only, while facial image can be reconstructed using both base and enhancement layer, by a generative decoder. While the main idea in [14] is very appealing, it is not clear how this approach can be extended to more general (e.g., non-face) image coding scenarios.

The approach presented in this paper builds on the idea of scalable latent-space representation, and is more widely applicable than [14]. In particular, it can accommodate generic learnt features and an arbitrary base task. For concreteness, our experiments are carried out on a model whose base task is object detection, but it should be noted that a similar methodology could be applied to another base task, such as image classification, object segmentation, etc.

### 3. PROPOSED METHOD

#### 3.1. Motivation

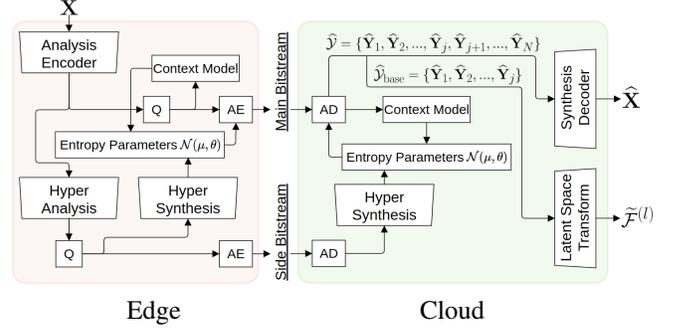
A Markov chain model for the CI system studied in this paper is shown in Fig. 1. Input image  $\mathbf{X}$  is processed by the edge sub-model  $f_1$ , producing features  $\mathcal{Y}$ . At the cloud side, from the features  $\mathcal{Y}$ , sub-model  $f_2$  reconstructs an approximation  $\hat{\mathbf{X}}$  to the input image  $\mathbf{X}$ , while sub-model  $f_3$  performs object detection, producing a collection  $T$  of bounding boxes and object classes.

Processing chain  $\mathbf{X} \rightarrow \mathcal{Y} \rightarrow \hat{\mathbf{X}}$  acts as an end-to-end image codec. Note that object detection can also be performed on the decoded image  $\hat{\mathbf{X}}$ , using an off-the-shelf object detector such as YOLO [16] or SSD [17], which is indicated by  $f_4$  in Fig. 1. In fact, such object detection from decoded images (rather than raw images) is common practice, because object detection datasets such as COCO [18] and ImageNet [19] contain JPEG-compressed images rather than raw images. Applying data processing inequality [20] to the Markov chain  $\mathcal{Y} \rightarrow \hat{\mathbf{X}} \rightarrow T$ , we have

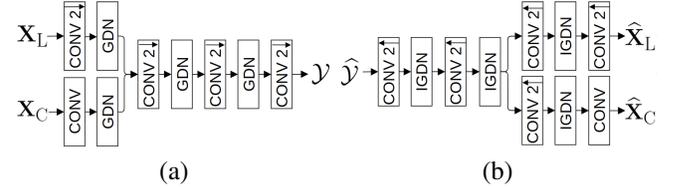
$$I(\mathcal{Y}; \hat{\mathbf{X}}) \geq I(\mathcal{Y}; T), \quad (1)$$

where  $I(\cdot; \cdot)$  denotes mutual information [20]. This suggests that intermediate features  $\mathcal{Y}$  carry less information about object detection ( $T$ ) than they do about input reconstruction ( $\hat{\mathbf{X}}$ ). This observation motivates our approach - we construct the features  $\mathcal{Y}$  such that only part of  $\mathcal{Y}$  is used for object detection, while the whole of  $\mathcal{Y}$  is used for input reconstruction.

Fig. 2 shows the architecture of our CI system. A number of modules in the system are based on [21], while the newly proposed modules are discussed in more detail below.



**Fig. 2.** Architecture of our CI system. ‘Q’ represents quantization, ‘AE’/‘AD’ represent arithmetic encoder/decoder. Configuration details of ‘Context Model’, ‘Entropy Parameters’ and ‘Hyper Analysis/Synthesis’ follow [21], whereas ‘Analysis Encoder’, ‘Synthesis Decoder’ and ‘Latent Space Transform’ are newly proposed for our use case.



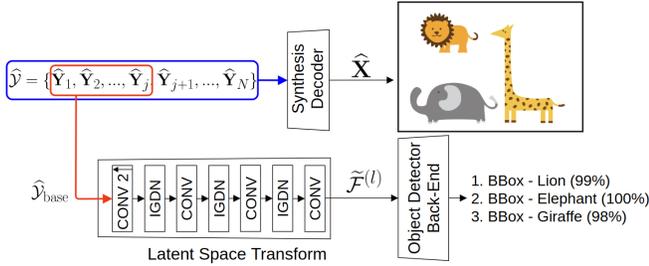
**Fig. 3.** Architecture of (a) Analysis Encoder and (b) Synthesis Decoder for YUV420 input/output

#### 3.2. Analysis Encoder and Synthesis Decoder

While most end-to-end learnt image compression methods [21–23] are made for RGB input images, we designed our system for YUV420 input format, which is more common in video coding. Specifically, input image  $\mathbf{X}$  comprises a luminance channel  $\mathbf{X}_L \in \mathbb{R}^{1 \times H \times W}$  and chrominance channels  $\mathbf{X}_C \in \mathbb{R}^{2 \times H/2 \times W/2}$ , where  $H \times W$  is the input resolution. The architectures of the corresponding Analysis Encoder and Synthesis decoder are shown in Fig. 3. The Analysis Encoder comprises a number of convolutional (‘CONV’) layers (with  $5 \times 5$  filters) and generalized divisive normalization (GDN) [24] layers. Downsampling in the luminance branch is performed by a convolution with stride 2. Synthesis Decoder is a mirror of the Analysis Encoder, with convolutions replaced by transpose convolutions (indicated by  $\uparrow$ ) and GDN layers replaced by inverse GDN (IGDN) layers. At the output of the Synthesis Decoder, the reconstructed input  $\hat{\mathbf{X}}$  consists of  $\hat{\mathbf{X}}_L$  and  $\hat{\mathbf{X}}_C$ .

#### 3.3. Latent-space scalability

The latent-space feature tensor in our system is of dimensions  $\mathcal{Y} \in \mathbb{R}^{N \times H/16 \times W/16}$ , consisting of  $N = 192$  channels:  $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_N\}$ . We split this tensor into two parts,  $\mathcal{Y}_{\text{base}} = \{\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_j\}$ , representing the



**Fig. 4.** Latent space transform to enable object detection from a subset of  $\hat{\mathcal{Y}}$

base-layer features with  $j < N$  channels, and  $\mathcal{Y}_{\text{enh}} = \{\mathbf{Y}_{j+1}, \mathbf{Y}_{j+2}, \dots, \mathbf{Y}_N\}$ , representing the enhancement-layer features with  $N - j$  channels. In our experiments, we used  $j = 128$ . At the decoder, if only object detection is required, only  $\mathcal{Y}_{\text{base}}$  needs to be reconstructed. If input image reconstruction is required, then the entire  $\mathcal{Y}$  is reconstructed.

An obvious question is - how do we know that the object detection-related information is concentrated in the first  $j$  channels of  $\mathcal{Y}$ ? This is achieved by training the entire model in Fig. 2 from scratch, as explained in Section 3.5. Through gradient-based updates from various loss terms, the model learns to steer the object detection-relevant information into  $\mathcal{Y}_{\text{base}}$ , while at the same time learning to reconstruct the input image using the entire  $\mathcal{Y}$ .

### 3.4. Latent space transform

We use the pre-trained back-end of YOLOv3 [16] for object detection in our system, specifically the portion from the batch normalization input in layer  $l = 12$  up to the model output. At this point, YOLOv3 expects a feature tensor  $\mathcal{F}^{(l)} \in \mathbb{R}^{256 \times H/8 \times W/8}$ , whereas our reconstructed base features are  $\hat{\mathcal{Y}}_{\text{base}} \in \mathbb{R}^{128 \times H/16 \times W/16}$ . Hence, a transformation from one latent space to another is needed. The structure of the latent space transform module is shown in Fig. 4; it consists of a transpose convolutional layer, whose purpose is to match the spatial resolution of the target latent space, and a sequence of IGDN and convolutional layers. At the output, a feature tensor in the target latent space,  $\tilde{\mathcal{F}}^{(l)}$ , is produced. Once  $\tilde{\mathcal{F}}^{(l)}$  is computed, it is fed to the batch normalization  $B^{(l)}$  of layer  $l = 12$  of YOLOv3, followed by  $\text{LeakyReLU}$  activation  $\sigma(\cdot)$ , thus producing the input to layer  $l = 13$ .

### 3.5. Training

Our loss function is in the form of a rate-distortion Lagrangian

$$\mathcal{L} = R + \lambda \cdot D, \quad (2)$$

where  $R$  is the rate estimate,  $D$  is the combined distortion for both input reconstruction and object detection, and  $\lambda$  is the Lagrange multiplier. Since our coding engine is based

on [21], the bitstream consists of the main bitstream, encoding latent data, and the side bitstream encoding the hyper-priors. Based on [21], rate estimates for these two bitstreams are

$$R = \underbrace{\mathbb{E}_{x \sim p_x} [-\log_2 p_{\hat{y}}(\hat{y})]}_{\text{main bitstream}} + \underbrace{\mathbb{E}_{x \sim p_x} [-\log_2 p_{\hat{z}}(\hat{z})]}_{\text{side bitstream}}, \quad (3)$$

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

$$\begin{aligned} D = & \text{MSE}(\mathbf{X}, \hat{\mathbf{X}}) \\ & + \alpha \cdot \text{MSE} \left( \sigma(B^{(l)}(\mathcal{F}^{(l)})), \sigma(B^{(l)}(\tilde{\mathcal{F}}^{(l)})) \right) \\ & + \beta \cdot (1 - \text{MS-SSIM}(\mathbf{X}, \hat{\mathbf{X}})), \end{aligned} \quad (4)$$

where  $\alpha$  and  $\beta$  are scale factors used to achieve various trade-offs, MSE is the mean squared error, and MS-SSIM is the multi-scale structural similarity index metric [25].

The first term in (4) encourages accurate reconstruction of the input image, while the third term encourages its perceptual quality. The second term is the MSE between the ground-truth feature tensor at the output of layer  $l = 12$  of YOLOv3 and the corresponding feature tensor derived from our base features  $\mathcal{Y}_{\text{base}}$ . Since this term depends only on  $\mathcal{Y}_{\text{base}}$  and not  $\mathcal{Y}_{\text{enh}}$ , gradients derived from it will update the model in such a way that the object detection-related information is steered towards  $\mathcal{Y}_{\text{base}}$ . Meanwhile, both  $\mathcal{Y}_{\text{base}}$  and  $\mathcal{Y}_{\text{enh}}$  contribute to input reconstruction, so the gradients derived from the first and third term in (4) will distribute input reconstruction-related information across both  $\mathcal{Y}_{\text{base}}$  and  $\mathcal{Y}_{\text{enh}}$ . Training of the system in Fig. 2 is carried out from scratch using a set of images  $\mathbf{X}$  and the corresponding ground-truth feature tensors obtained at the output of layer  $l = 12$  of YOLOv3 for those images.

## 4. EXPERIMENTS

Our model was trained on CLIC [26] and JPEG-AI [27] datasets. Images from the JPEG-AI dataset were resized to  $1920 \times 1080$  using the Lanczos filter. From the CLIC dataset, only images having resolutions  $320 \times 320$  or larger were used. Images were cropped using a random window with size of  $256 \times 256$  during training. Ground-truth feature tensors at layer 12 of YOLOv3 were generated for all training images, to enable computing the second term in (4). During training, these tensors were cropped to  $256 \times 32 \times 32$ , to match the position of the random  $256 \times 256$  window in the input image. Adam optimizer with a learning rate of  $10^{-4}$  was used to train the network for 2M epochs on a GeForce RTX 2080 GPU with 11 GB RAM. Similarly to [21], one model is trained for each  $\lambda \in \{0.005, 0.01, 0.02, 0.05, 0.1, 0.2\}$  in (2).

Since our model supports two tasks, we compare it against relevant benchmarks for each task. For object detection, the model is evaluated on the COCO 2014 validation dataset, which includes about 5K JPEG-compressed images. The average file size of these JPEG images is around 1260 Kbits, and

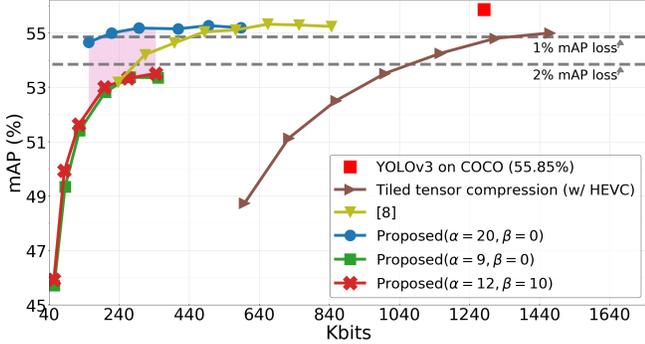


Fig. 5. Object detection performance of various methods.

off-the-shelf YOLOv3 achieves the mean Average Precision (mAP) of 55.85% on these images. This is shown as the red square in Fig. 5. When HEVC-Intra is used to encode tiled, 8-bit pre-quantized tensors from layer 12 of YOLOv3, the brown curve in Fig. 5 is obtained. Our recent work [8], which we believe is state-of-the-art for coding YOLOv3 feature tensors, is shown as the yellow curve. To evaluate the proposed model, input images were first converted to YUV420 using `ffmpeg`, then fed to the model. Performance curves for several values of  $\alpha$  and  $\beta$  in (4) are shown as red, green, and blue in Fig. 5. Note that increasing  $\alpha$  improves object detection performance, as expected from (4). With  $\alpha = 20$ , outstanding performance on object detection can be achieved at very low bitrates, with less than 1% mAP loss compared to default YOLOv3. Even with lower  $\alpha$ , the proposed model is competitive with [8] at low bitrates. The shaded area in Fig. 5 shows the operating range that can be achieved by varying  $\alpha$ .

When input reconstruction is used, our model acts as an end-to-end image codec, so we compare it against relevant benchmarks on raw YUV420 HEVC common test sequences [28]. One benchmark is HEVC (HM-16.20) [29] with All Intra configuration [28]. Since the backbone of our model is based on [21], we use the model from [21] as the second benchmark. To encode YUV420 input using [21], chrominance channels were upsampled using nearest-neighbor interpolation and then converted to RGB using `ffmpeg`. RGB output was converted back to YUV420 using `ffmpeg`.

Table 1 shows the average BD-Bitrate against HEVC, computed over Y-PSNR vs. bits curves. Our model with  $(\alpha, \beta) = (12, 10)$  shows better performance than [21] on sequences in classes A, B, and C, and even better than HEVC in class A. This can be expected from (4), since smaller  $\alpha$  together with larger  $\beta$  de-emphasizes object detection performance and in turn promotes input reconstruction. Meanwhile, setting  $(\alpha, \beta) = (20, 0)$  leads to a considerable loss in input reconstruction efficacy, but in turn achieves outstanding object detection performance, as seen in Fig. 5.

Table 2 shows the average BD-Bitrate against HEVC, computed over Y-MS-SSIM vs. bits curves. It is well known that end-to-end deep model-based image codecs perform well

Table 1. BD-Bitrate (Bits vs. Y-PSNR) vs. HM-16.20

Class	Baseline	Proposed method		
	[21]	$(\alpha = 9, \beta = 0)$	$(\alpha = 12, \beta = 10)$	$(\alpha = 20, \beta = 0)$
A	8.54%	-2.60%	<b>-6.93%</b>	150.48%
B	43.91%	9.08%	<b>3.16%</b>	206.82%
C	17.98%	22.17%	<b>14.57%</b>	198.05%
D	<b>17.14%</b>	28.99%	22.25%	188.54%

Table 2. BD-Bitrate (Bits vs. Y-MS-SSIM) vs. HM-16.20

Class	Baseline	Proposed method		
	[21]	$(\alpha = 9, \beta = 0)$	$(\alpha = 12, \beta = 10)$	$(\alpha = 20, \beta = 0)$
A	<b>-27.47%</b>	-17.11%	-23.72%	45.64%
B	-10.61%	-9.09%	<b>-16.25%</b>	79.31%
C	<b>-39.00%</b>	-3.88%	-10.80%	80.55%
D	<b>-27.84%</b>	0.98%	-5.32%	77.14%

on MS-SSIM, and this is indeed noticeable in Table 2. Here, the model from [21] outperforms HEVC in all sequence classes, and our model with  $(\alpha, \beta) = (12, 10)$  does so as well. Our model does better than [21] in class B, while in other classes, [21] offers better performance. It can also be noted that loss of input reconstruction efficacy of our model with  $(\alpha, \beta) = (20, 0)$  is now much smaller when measured against MS-SSIM.

Overall, the above results show that the proposed model can achieve comparable compression efficiency to [21] when used as an end-to-end image codec, and can be better than HEVC when the reconstruction quality is measured using MS-SSIM. On top of that, our model offers scalability to perform object detection from a subset of the latent space, which neither [21] nor HEVC (nor any other codec, to our knowledge) is currently able to offer. We therefore believe it will be a useful contribution to future research and standardization activities in this area.

## 5. CONCLUSIONS

We introduced latent-space scalability for multi-task collaborative intelligence, and tested it on a system that supports object detection and input reconstruction. A part of the latent space is dedicated to object detection, while the whole latent space is used for input reconstruction. Appropriately chosen loss terms allow for steering relevant information to different portions of the latent space as the model is trained. By varying the scaling factors of various loss terms, different trade-offs between the two tasks were demonstrated and compared with the relevant benchmarks.

## 6. REFERENCES

- [1] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. N. Mudge, J. Mars, and L. Tang, “Neurosurgeon: Collaborative intelligence between the cloud and mobile edge,” in *ASPLOS '17*, Apr. 2017.
- [2] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, “JointDNN: an efficient training and inference engine for intelligent mobile cloud computing services,” *IEEE Trans. Mobile Computing*, vol. 20, no. 2, pp. 565–576, Feb. 2021.
- [3] “Call for evidence for video coding for machines,” ISO/IEC JTC 1/SC 29/WG 2, m55065, Oct. 2020.
- [4] J. Ascenso, “JPEG AI use cases and requirements,” ISO/IEC JTC 1/SC29/WG1 M90014, Jan. 2021.
- [5] H. Choi and I. V. Bajić, “Deep feature compression for collaborative object detection,” in *Proc. IEEE ICIP*, Oct. 2018, pp. 3743–3747.
- [6] A. E. Eshratifar, A. Esmaili, and M. Pedram, “Towards collaborative intelligence friendly architectures for deep learning,” *20th Int. Symposium Quality Electronic Design (ISQED)*, pp. 14–19, Mar. 2019.
- [7] A. E. Eshratifar, A. Esmaili, and M. Pedram, “BottleNet: A deep learning architecture for intelligent mobile cloud computing services,” in *Proc. IEEE/ACM Int. Symposium Low Power Electronics and Design (ISLPED)*, Jul. 2019.
- [8] H. Choi, R. A. Cohen, and I. V. Bajić, “Back-and-forth prediction for deep tensor compression,” in *Proc. IEEE ICASSP*, May 2020, pp. 3743–3747.
- [9] R. A. Cohen, H. Choi, and I. V. Bajić, “Lightweight compression of neural network feature tensors for collaborative intelligence,” in *Proc. IEEE ICME*, Jul. 2020.
- [10] H. Choi and I. V. Bajić, “Near-lossless deep feature compression for collaborative intelligence,” in *Proc. IEEE MMSP*, Aug. 2018.
- [11] 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'18*, 2018.
- [12] 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.
- [13] L. Duan, J. Liu, W. Yang, T. Huang, and W. Gao, “Video coding for machines: A paradigm of collaborative compression and intelligent analytics,” *IEEE Transactions on Image Processing*, vol. 29, pp. 8680–8695, 2020.
- [14] 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 *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.
- [15] X. Zhang, S. Ma, S. Wang, X. Zhang, H. Sun, and W. Gao, “A joint compression scheme of video feature descriptors and visual content,” *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 633–647, 2016.
- [16] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, Apr. 2018.
- [17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [18] 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 *European Conf. on Computer Vision (ECCV)*, Sept. 2014.
- [19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- [20] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 2nd edition, 2006.
- [21] 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.
- [22] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior,” in *International Conference on Learning Representations*, 2018.
- [23] J. Ballé, V. Laparra, and E. Simoncelli, “End-to-end optimized image compression,” in *5th International Conference on Learning Representations, ICLR 2017*, 2019.
- [24] J. Ballé, V. Laparra, and E. P. Simoncelli, “Density modeling of images using a generalized normalization transformation,” *arXiv preprint arXiv:1511.06281*, 2015.
- [25] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*. Ieee, 2003, vol. 2, pp. 1398–1402.
- [26] “Challenge on learned image compression (CLIC),” [Online]: <http://www.compression.cc/>.
- [27] “JPEG AI dataset,” [Online]: <https://jpeg.org/jpegai/dataset.html>.
- [28] F. Bossen, “Common HM test conditions and software reference configurations,” in *ISO/IEC JTC1/SC29 WG11, JCTVC-L1100*, Jan. 2013.
- [29] 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.