

IMAGE CODING FOR MACHINES: AN END-TO-END LEARNED APPROACH

Nam Le*, Honglei Zhang[†], Francesco Cricri[†], Ramin Ghaznavi-Youvalari[†], Esa Rahtu*

[†]Nokia Technologies, *Tampere University
Tampere, Finland

ABSTRACT

Over recent years, deep learning-based computer vision systems have been applied to images at an ever-increasing pace, oftentimes representing the only type of consumption for those images. Given the dramatic explosion in the number of images generated per day, a question arises: how much better would an image codec targeting machine-consumption perform against state-of-the-art codecs targeting human-consumption? In this paper, we propose an image codec for machines which is neural network (NN) based and end-to-end learned. In particular, we propose a set of training strategies that address the delicate problem of balancing competing loss functions, such as computer vision task losses, image distortion losses, and rate loss. Our experimental results show that our NN-based codec outperforms the state-of-the-art Versatile Video Coding (VVC) standard on the object detection and instance segmentation tasks, achieving -37.87% and -32.90% of BD-rate gain, respectively, while being fast thanks to its compact size. To the best of our knowledge, this is the first end-to-end learned machine-targeted image codec.

Index Terms— image coding for machines, image compression, loss weighting, multitask learning, video coding for machines

1. INTRODUCTION AND BACKGROUND

Over the years, traditional image and video coding standards such as Versatile Video Coding (VVC) [1] have significantly improved the coding efficiency or rate-distortion performance, in which the “distortion” is measured by metrics aimed at improving the quality for human consumption. In the era of deep learning, computer vision tasks such as object detection or image classification represent a significant portion of image and video consumers, oftentimes being the only ones, e.g. self-steering system for cars in [2]. In this paper, we refer to computer vision tasks simply as *machines*. These *machines* are usually applied (and trained) on human-targeted compressed images, which contain distortions that are less perceivable by humans but may lower the performance of the computer vision tasks. In addition, a lot of information contained in these compressed images is not necessary for a neural network (NN) to perform a task, such

as detecting objects. Thus, it is likely that higher coding efficiency can be achieved by designing a codec with the specific goal of targeting *machines* as the only consumer.

In order to directly improve the task performance, [3] and [4] propose standard-compliant methods that preserve the standard coded bitstream format by fine-tuning specific parts of the traditional codec for the targeted *machines*. Although the above methods manage to improve the task performance, they do not aim to completely replace the conventional pipeline for human-oriented coding, instead they only add an incremental capability to the system. The use of neural networks to aid or completely replace the traditional image codec for human consumption has been actively studied recently. In [5], a CNN post-processing filter is used to enhance the image quality. Some other proposals seek to effectively model the distribution of the data that is encoded/decoded by a lossless codec. To this end, [6] proposes an autoregressive scheme for distribution modeling, while the system in [7] hierarchically learns the distribution through feature maps at multiple scales. In [8, 9], the authors propose end-to-end pipelines for image *transform coding*, where the image’s latent representation and its distribution are entropy encoded using hierarchical learned *hyperprior*. Our work can be viewed as an extension of [9], in which the main targeted end-users are *machines*, and the main coder is substituted by a more capable one. Regarding the problem of multi-objective training, [10] discusses the importance of loss weighting and methods for dynamic loss balancing, as opposed to fixed weighted losses. These methods, however, do not directly offer full control over the priorities of the objectives, which is a desired feature in our case.

In this paper, we propose an end-to-end learned system that directly optimizes the rate-distortion trade-off, where the distortion is the training loss of the pretrained task-NN. In order to achieve better coding efficiency than state-of-the-art traditional codecs, we introduce an adaptive loss weighting strategy addressing the problem of balancing competing losses in multi-task learning. In our experimental section, we test our proposed techniques on the tasks of object detection and instance segmentation, and show that our system outperforms VVC significantly.

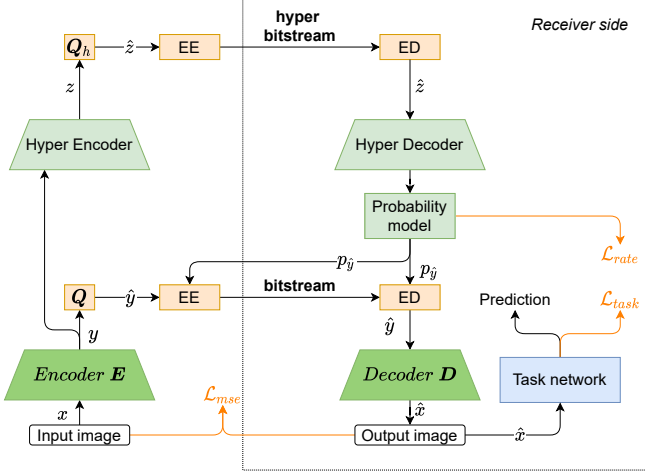


Fig. 1. Image Coding for Machines overview. “EE” and “ED” denote entropy encoders and decoders, respectively.

2. PROPOSED METHOD

In contrast to the pipeline in [9], we propose an Image Coding for Machines (ICM) system targeting task performance instead of pixel-domain fidelity. Our codec comprises a neural auto-encoder, a learned probability model and an entropy codec. The proposed pipeline is illustrated in Figure 1. The NN-based encoder transforms the uncompressed image x to a new data representation $y = E(x; \theta_E)$, which is then quantized as $\hat{y} = Q(y)$ and subsequently lossless-compressed by an entropy encoder, using the probability distribution estimated by the probability model. The output bitstream is decompressed on the decoder-side and decoded back to the pixel domain by the NN decoder as $\hat{x} = D(\hat{y}; \theta_D)$. The task NN takes \hat{x} as input and returns the corresponding task results.

2.1. Auto-encoder

Unlike common auto-encoders, our proposed auto-encoder does not aim to reconstruct the input image. Instead, its goal is to decode a data tensor that can provide a good task performance, while the encoder’s output can be efficiently compressed by the entropy codec. These two objectives are referred to as task loss \mathcal{L}_{task} and rate loss \mathcal{L}_{rate} , respectively. As the task NNs are already pretrained and left unmodified, they accept input data in the format of images, i.e., three channels. Thus, the output of the decoder needs to be a tensor of same shape as an image. For the architecture of the encoder and decoder, we use a convolutional neural network (CNN) architecture with residual connections, as illustrated in Figure 2. In order to keep encoding-decoding time and resource consumption low, we chose to use CNNs with a small number of filters in the intermediate and last layers. This auto-encoder is optimized by using the aforementioned loss terms.

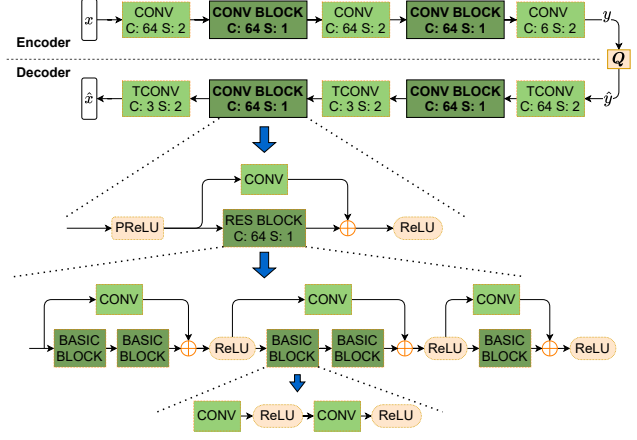


Fig. 2. Auto-encoder architecture. The convolutional blocks are illustrated by sharp rectangles. “TCONV” denotes the transposed convolutional layers. In each convolutional block, “S” denotes the stride and “C” denotes number of output channels for all of the children blocks. These values are inherited from the parent block if not stated otherwise.

2.2. Probability model

An asymmetric numeral systems (ANS) [11] codec first encodes a stream of symbols into a single natural number according to the probability of each symbol, then it converts the number into a binary bitstream. Given the quantized latent tensor \hat{y} and its estimated distribution $p_{\hat{y}}(\hat{y})$, we’re interested to encode the symbols of \hat{y} with a minimum code length lower-bounded by Shannon entropy. The lower-bound is achieved if the marginal distribution $m_{\hat{y}}$ is identical to $p_{\hat{y}}$. Since $m_{\hat{y}}$ arises from the unknown input image distribution p_x and the transformation method $Q(E(\cdot; \theta_E))$, the code length r can only be estimated by Shannon cross-entropy:

$$\begin{aligned} r &= \mathbb{E}_{\hat{y} \sim m_{\hat{y}}} [-\log_2 p_{\hat{y}}(\hat{y})] \\ &= \mathbb{E}_{x \sim p_x} [-\log_2 p_{\hat{y}}(Q(E(x; \theta_E)))] \end{aligned} \quad (1)$$

The probability model aims to learn the distribution $p_{\hat{y}}$ in order to minimize r . For this module, we use the “Mean & Scale Hyperprior” structure proposed in [9] which models the latent by Gaussian mixture models and learns the parameters to these distributions. The distribution $p_{\hat{y}}$ is obtained on-the-fly using these parameters in a closed-form fashion. In order to decompress the coded latents \hat{y} , an additional bitstream for the “hyper-latents” \hat{z} carrying the hyperprior embeddings is sent to the receiver side. The rate loss term \mathcal{L}_{rate} is thus given by the total length of the two bitstreams:

$$\mathcal{L}_{rate} = \underbrace{\mathbb{E}_{\hat{y} \sim m_{\hat{y}}} [-\log_2 p_{\hat{y}}(\hat{y})]}_{\text{latents rate}} + \underbrace{\mathbb{E}_{\hat{z} \sim m_{\hat{z}}} [-\log_2 p_{\hat{z}}(\hat{z})]}_{\text{hyper-latents rate}} \quad (2)$$

The probability model is jointly optimized with the auto-encoder in an end-to-end training fashion. During training, the quantization step is replaced by additive uniform noise to make gradient-based optimization possible [8, 12].

2.3. Training strategy

We separately trained and evaluated two different compression models for two computer vision tasks: object detection, using Faster R-CNN [13], and instance segmentation, using Mask R-CNN [14]. In each case, we freeze the corresponding pre-trained task network and define \mathcal{L}_{task} as the respective training task loss. Thus, gradients of \mathcal{L}_{task} are computed only with respect to the codec’s parameters.

Image coding is often posed as a rate-distortion optimization (RDO) problem, i.e. $J = R + \lambda \cdot D$, where R, D and J denote the bitrate, the distortion and joint cost, respectively, and λ is the Lagrange multiplier driving the trade-off between them: more encoding bits (high R) reduces the distortion (low D) and vice versa [1, 7, 9]. The common way of performing RDO when training NN-based codecs is to find a certain set of values for λ (and other hyper-parameters) so that the desired rate-distortion is achieved after a number of training iterations [7, 9]. The learned models are then saved as compression models to be used for the corresponding bitrates. Given that the decoded images are consumed by machines, it is not necessary to have high fidelity output images that are visually appealing to humans. We instead prioritize good task performance by imposing task loss \mathcal{L}_{task} minimization on our model training. Our ICM system extends the above RDO approach by adding the task loss \mathcal{L}_{task} to the “distortion” D . The general training loss function is given by:

$$\mathcal{L}_{total} = w_{rate}\mathcal{L}_{rate} + w_{mse}\mathcal{L}_{mse} + w_{task}\mathcal{L}_{task}, \quad (3)$$

where $w_{rate}, w_{mse}, w_{task}$ are the scalar weights for each loss term $\mathcal{L}_{rate}, \mathcal{L}_{mse}, \mathcal{L}_{task}$, respectively. $\mathcal{L}_{mse} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2$, where N denotes the mini-batch size. \mathcal{L}_{task} and \mathcal{L}_{rate} were defined earlier in this and the previous subsections.

Loss weighting strategy: Rather than using fixed loss weights, we propose a dynamic loss weighting strategy for an effective multitask training because: *i)* The competing nature of the loss terms makes their respective gradients to worsen the performance of the others’. It is critical to have a right balance between the objectives in each update, which is very challenging for fixed loss weighting due to its inflexibility. *ii)* Exhaustive search for the optimal weights is very time-consuming [15].

The task networks are trained on natural images, thus expect close-to-natural images as their input. In that light, we train a base model with only \mathcal{L}_{mse} ($w_{task} = w_{rate} = 0, w_{mse} = 1$), which is capable of reconstructing images for a decent task performance. Then we fine-tune the base model by gradually raising w_{rate} and w_{task} in different phases while keeping $w_{mse} = 1$ as shown in Figure 3, which eventually

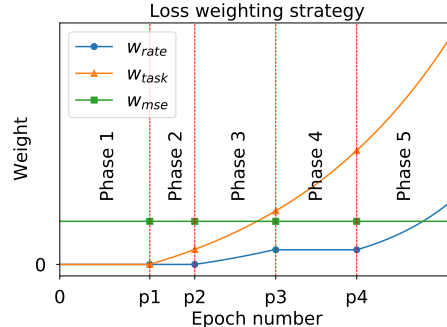


Fig. 3. Loss weights evolution over iterations. There are 5 phases in this strategy, separated by vertical lines. Phase 1-5 respectively: base model training with \mathcal{L}_{mse} , introducing \mathcal{L}_{task} , introducing \mathcal{L}_{rate} , enhancing task performance, searching for optimal trade-offs when the system is stable.

leads to the dominant impact of the gradients of \mathcal{L}_{rate} and \mathcal{L}_{task} on the accumulated gradients flow, effectively pushing the system to achieve an optimal task performance for a given bitrate constraint. Consequently, the same training instance is able to achieve a new rate-distortion performance for a different targeted bitrate after every iteration (see the results in Figure 4). Learning rate decay is applied to keep the training stable. At inference time, a desired bitrate can be achieved by using the closest model’s checkpoint (i.e., saved parameters) in terms of bitrate achieved during training on a validation set.

3. EXPERIMENTS AND DISCUSSION

3.1. Experimental setup

The framework in section 2 is evaluated on two tasks: instance segmentation and object detection. The pre-trained models are provided by Torchvision¹, and the rest of the framework is implemented with Pytorch 1.5. We use the uncompressed dataset Cityscapes [16] for training and testing our proposed models. Since the task models are pre-trained on COCO dataset [17], we only evaluate results on the classes that are common between the two datasets: *car, person, bicycle, bus, truck, train, motorcycle*.

Evaluation method and baseline: We evaluated the performance of each compression method based on its average bitrate and task performance over 500 images of the *val* set. We use bits per pixel (BPP) as the bitrate metric and Mean Average Precision (mAP@[0.5:0.05:0.95])[16] for the task performance.

As our baseline, we use the current state-of-the-art codec standard VVC (reference software VTM-8.2 [18], All-Intra configuration), under JVET common test conditions (CTC)

¹The pre-trained models can be found at <https://pytorch.org/docs/stable/torchvision/models.html>

[19]. In order to achieve different bitrates, we encoded the *val* set using 28 settings, which are the combinations of 7 quantization parameters (QP) (22, 27, 32, 37, 42, 47, 52) and 4 downsampling factors (100% which corresponds to original resolution, 75%, 50% and 25%). This results in 28 coded versions of the validation dataset. For the downsampled versions, data are upsampled to the original resolution before given to the task networks for task performance evaluation.

Loss weighting: In these experiments, we model the loss weight values as functions of the epoch number. Concretely, the strategy shown in Figure 3 is formulated by

$$\begin{aligned}
 w_{mse} &= 1, \\
 w_{task} &= \begin{cases} 0, & e < p_1 \\ 4f_w(e - p_1, 1.01), & e \geq p_1 \end{cases}, \\
 w_{rate} &= \begin{cases} 0, & e < p_2 \\ 2f_w(e - p_2, 1.01), & p_2 \leq e < p_3 \\ c, & p_3 \leq e < p_4 \\ c + 2f_w(e - p_4, 1.02), & e \geq p_4 \end{cases}, \quad (4)
 \end{aligned}$$

where $p_1 = 50, p_2 = 75, p_3 = 120, p_4 = 165, f_w(x, a) = 10^{-3}(a^x - 1)$ and $c = 2f_w(p_3 - p_2 - 1, 1.01)$.

3.2. Experimental results

The Rate-Performance curves of our two systems against the corresponding baselines are shown in Figure 4. After every epoch of training set, we evaluated our models on *val* set and obtained one data point for the curve “Our method”. As the figure shows, our systems achieves superior performance for both tasks. Interestingly, both of the systems can achieve higher mAP than the best quality VVC settings (QP 22, 100% resolution) while consuming only around 60% of the bitrate. As shown in Table 1, our method on average saves 37.87% bitrate for the same task performance level on object detection and 32.90% on instance segmentation, in comparison with the Pareto front of VVC anchors. As a reference, although not directly comparable, for instance segmentation using the same task-NN architecture (Mask R-CNN) but trained on CityScapes *train* set, [3] reports up to 9.95% of bitrate saving for the following QPs: 12, 17, 22, 27, on the CityScapes *val* set. Our system contains only 1.5M parameters, and is also extremely fast: the average encoding time for a 2048×1024 image in the *val* set is around 0.15 seconds, with batch size of 1 on a single RTX 2080Ti GPU².

The reconstructed outputs in Figure 5 indicate that the system learns to compress more aggressively on the regions that are not too important to the task network. In particular, in low bitrate settings, the shapes and edges of the objects are preserved for the purpose of the machine tasks. Although the systems are trained on specific task network architectures, it

²As a reference, encoding a 2048×1024 image by VTM-8.2 software takes about 125 seconds on a cluster with Intel Xeon Gold 6154 CPUs.

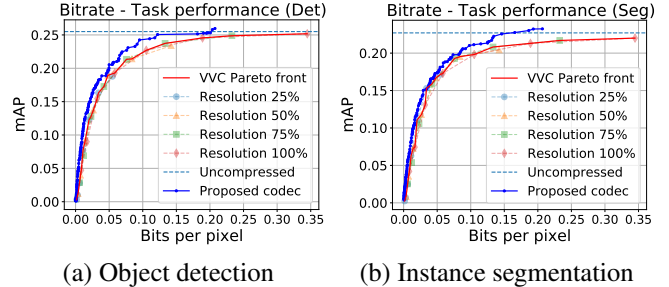


Fig. 4. Rate-Performance curves. Each point on the dashed lines represents the performance of a VVC coded versions of the data with a certain QP and resolution combination.

Table 1. Bjøntegaard Delta rate (BD-rate) with respect to task performance against VVC anchors as in Figure 4.

	100%	75%	50%	25%	Pareto
Detection	-32.86	-33.89	-34.76	-38.76	-37.87
Segmentation	-33.77	-28.58	-29.98	-28.04	-32.90

preserves critical information from the input data that may be suitable for different tasks or different NN architectures, and this is a subject for our future work.

4. CONCLUSIONS

We proposed and evaluated an efficient and fast end-to-end trained system for ICM. It shows that the proposed NN-based codec outperforms the state-of-the-art traditional codec VVC by a large margin when the compression is for machines. Additionally, we introduced a flexible loss weighting strategy for multi-task learning that both effectively trains the model towards the desired objectives and achieves optimal trade-off between them along the way at the same time.



Fig. 5. Decoded outputs for instance segmentation in different bitrates. The background is fiercely suppressed in low bitrate.

5. REFERENCES

- [1] B. Bross, J. Chen, S. Liu, and Y.-K. Wang, “Versatile Video Coding (draft 8),” *Joint Video Experts Team (JVET), Document JVET-Q2001*, January 2020.
- [2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016.
- [3] K. Fischer, F. Brand, C. Herglotz, and A. Kaup, “Video coding for machines with feature-based rate-distortion optimization,” *IEEE 22nd International Workshop on Multimedia Signal Processing*, p. 6, September 2020.
- [4] B. Brummer and C. de Vleeschouwer, “Adapting JPEG XS gains and priorities to tasks and contents,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. pp. 629–633, IEEE.
- [5] Z. Wang, R.-L. Liao, and Y. Ye, “Joint learned and traditional video compression for p frame,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. pp. 560–564, IEEE.
- [6] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” New York, New York, USA, 20–22 Jun 2016, vol. 48 of *Proceedings of Machine Learning Research*, pp. 1747–1756, PMLR.
- [7] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, “Practical full resolution learned lossless image compression,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [8] 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.
- [9] D. Minnen, J. Ballé, and G. D. Toderici, “Joint autoregressive and hierarchical priors for learned image compression,” in *Advances in Neural Information Processing Systems 31*, pp. 10771–10780. Curran Associates, Inc., 2018.
- [10] T. Gong, T. Lee, C. Stephenson, V. Renduchintala, S. Padhy, A. Ndirango, G. Keskin, and O. H. Elibol, “A comparison of loss weighting strategies for multi task learning in deep neural networks,” *IEEE Access*, vol. 7, pp. 141627–141632, 2019.
- [11] J. Duda, K. Tahboub, N. J. Gadgil, and E. J. Delp, “The use of asymmetric numeral systems as an accurate replacement for huffman coding,” in *2015 Picture Coding Symposium (PCS)*, pp. 65–69.
- [12] J. Ballé, V. Laparra, and E. P. Simoncelli, “End-to-end optimization of nonlinear transform codes for perceptual quality,” in *2016 Picture Coding Symposium (PCS)*, pp. 1–5, ISSN: 2472-7822.
- [13] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems 28*, pp. 91–99. Curran Associates, Inc., 2015.
- [14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, ISSN: 2380-7504.
- [15] R. Cipolla, Y. Gal, and A. Kendall, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7482–7491, ISSN: 2575-7075.
- [16] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 3213–3223, IEEE.
- [17] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft COCO: Common objects in context,” 2015.
- [18] “Versatile video coding (VVC) reference software VTM-8.2,” Available at: https://vcgit.hhi.fraunhofer.de/jvetVVCSoftware_VTM (Accessed on 2020-10-20).
- [19] F. Brossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, “JVET common test conditions and software reference configurations for sdr video,” *Joint Video Experts Team (JVET), Document: JVET-N1010*, March 2019.