Tampere University

Nam Le

Still image coding for machines

An end-to-end learned approach

Faculty of Information Technology and Communication Sciences (ITC)
Master's thesis
Supervisor: Professor Esa Rahtu
November 2020

Abstract

Nam Le: Still image coding for machines
Master's thesis
Tampere University
In collaboration with the Future Video Coding team, Nokia Technologies, Finland.
Master's Degree Programme in Data engineering and machine learning
November 2020

The ever-increasing pace of neural network (NN) based solutions for computer vision tasks is making them one of the main consumers of digital images nowadays. This raises the question of whether the traditional human-oriented image codecs, or the adapted version of these codecs for the machine-targeted use cases are efficient enough for the massive amount of image data generated every day for both humans and machines. This thesis explores the abilities of the image codecs that are designed specifically only for machine-consumption. To the best of the student's knowledge, this is the first end-to-end learned machine-oriented image codec proposal. It presents an end-to-end framework for designing NN-based image codecs for machines, as well as a set of training strategies that address the delicate problem of balancing competing losses in multi-task training, namely image distortion loss, rate loss, and computer vision task losses. The experimental results show the superior coding efficiency of the proposed codecs in comparison with the current state-of-the-art standard VVC/H.266 on object detection and instance segmentation, achieving -37.87% and -32.90% BD-rate gain, respectively while being extremely fast thanks to its compact size. These results also serve as a proof-of-concept for a new approach to Image coding for machines.

Keywords: image coding for machines, loss weighting, multitask training, deep learning, image codec.

Contents

1 Introduction

For the last decades, digitalized signals have been replacing the role of analog signals as the main form for media archiving. Video coding describes the techniques of storing digitalized video signal (encoding), and reconstructing the input data from the encoded data (decoding). The primitive objective of video coding is to reduce the footprint of the encoded data while preserving the fidelity of the decoded data, therefore video coding is also referred to as video compression. There are mainly two groups of coding techniques: lossless compression and lossy compression. Lossless compression techniques seek to faithfully restore the coded data with perfect fidelity to the original input, while the lossy techniques explore the optimal trade-offs between footprint size and the volume of distortions in the reconstructed data with respect to its input. This thesis will only discuss the latter approach, and such devices or programs that encode or decode data streams are hereafter referred to as "codecs".

Image coding is a special case of video coding where the video sequence contains only one frame, hence the development of video coding also implies the evolution of image coding.

1.1 Traditional coding standards - image coding for humans

There are many video coding standards developed throughout the years, such as AVC/H.264 [1], HEVC/H.265 [2], VVC/H.266 [3] and VP8 [4]. These standards can be used for image coding with the "All-Intra" configuration, where each frame in the video sequences is coded independently. In other words, in this configuration, each frame is treated as an individual image. Within the scope of this thesis, video codecs in the "All-Intra" configuration are considered as image codecs and they are referred to only in this configuration. There are also other image coding standards that are designed specifically for image coding, such as JPEG [5], JPEG-2000 [6, 7], WebP [8] and BPG [9].

All of the above-mentioned standards exploit the human vision redundancies to achieve lower bitrate. Figure 1.1 illustrates the pipeline of image coding for human consumption and the trade-off optimization problem that comes with it. In this pipeline, the image visual quality is measured by human evaluation, which is why the traditional standards are always optimized for the human visual system.

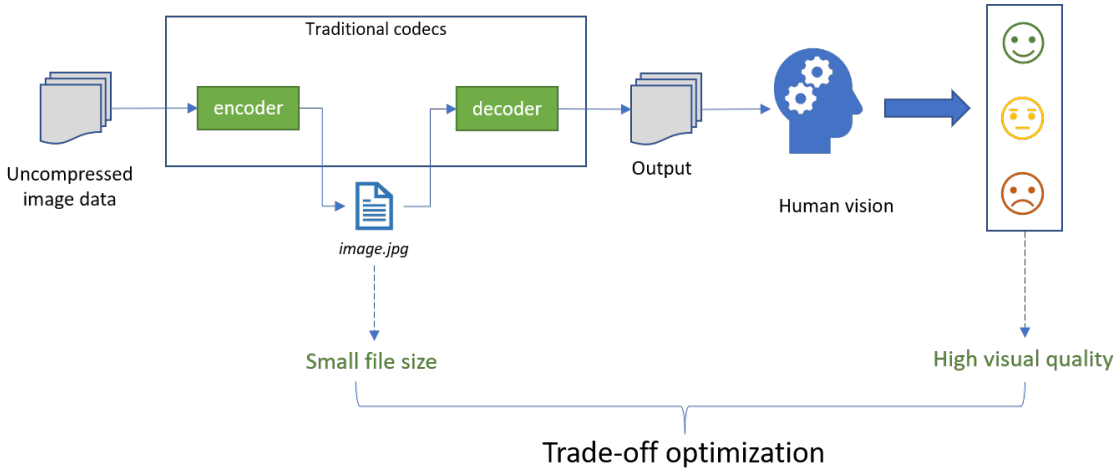


Figure 1.1 *The traditional approach of image coding – Image coding for humans. In this scheme, the uncompressed image data is compressed and decompressed respectively by the encoder and decoder from the traditional codecs, such as HEVC or VVC. The decompressed data is evaluated in terms of visual quality by human consumers. The coding efficiency is measured considering the bitstream size or bitrate and the visual quality.*

1.2 Image coding for machines

According to Cisco Annual Internet Report (2018-2023) [10], machine-to-machine connections will be the fastest-growing category, taking up 50% of the total devices and connections by 2023. The rising use of neural network (NN)-based approaches for computer vision tasks (Figure 1.2) has made machine learning applications, besides humans, occupy a large proportion of image data consumption, oftentimes being the only ones, e.g. self-steering system for autonomous cars in [11]. In this thesis the computer vision tasks are simply referred to as *machines*. These *machines* are usually trained and applied on human-targeted compressed images. The problems with this approach are two-fold:

- Machine vision does not share all the characteristics of human vision. The human-targeted images are compressed by the traditional coding standards that exploit the human visual redundancies, hence may contain distortions that are less perceivable by humans but can severely damage the performance of the *machines*.
- A significant amount of the information contained in these images is likely to be unnecessary for a NN to perform a task. For example, a pedestrian detection network does not need too many details of the sky or of the buildings on the sides of the road.

These problems directly lead to sub-optimal compression performance. Thus, it is likely that higher coding efficiency can be achieved with a codec that is designed

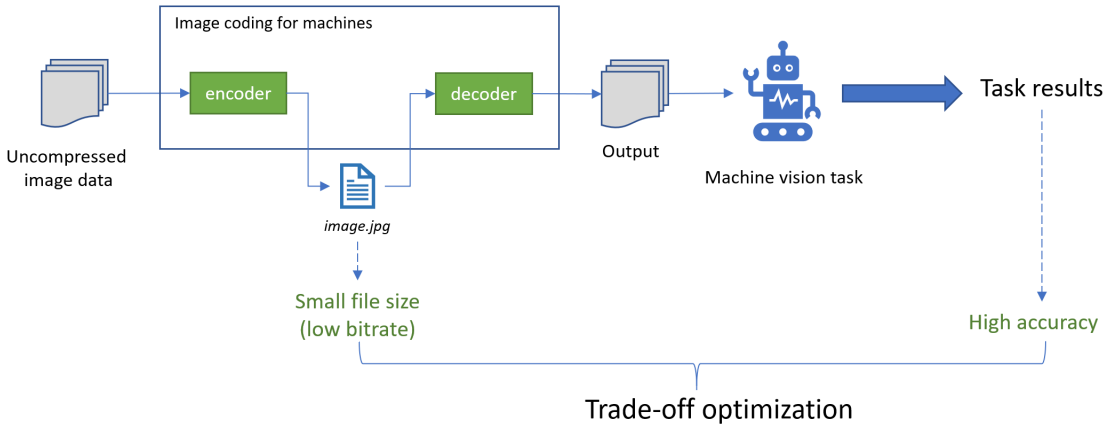


Figure 1.2 *Image coding for machines – ICM. Instead of optimizing for human visual quality, the ICM codecs aim for vision task accuracy. The coding efficiency therefore is measured by bitrate–task performance trade-off optimization.*

to serve *machines* as its first-class consumer. Additionally, machine learning applications usually require a massive amount of data for their model training. Therefore a new domain of solutions for image coding specifically designed for machine-consumption is desirable. This new category of image coding techniques is expected to enhance data compression efficiency while preserving machines' performance, consequently named "Image coding for machines" (ICM).

The ultimate goal of ICM is to have superior coding efficiency when the consumers are *machines*, without the knowledge of its down-stream tasks, therefore can effectively replace the role of traditional image coding standards in these use cases. This is an emerging topic and it has been actively studied in recent years, alongside Video coding for machines (VCM). An overview of the challenges and prior work to this problem will be presented in chapter 2. At this early stage, while being completely task-agnostic is still the long-term target, the current studies, including this thesis, also explore the potential of different coding paradigms that are conditioned to some extent to the information of the consumer *machines*.

In the scope of this thesis, a new end-to-end approach to ICM is proposed, its effectivity is explored as a proof-of-concept on two task networks: object detection and instance segmentation, and its potential is discussed in the later chapters. This thesis views the two main goals of image coding for machines, low bitrate and high task performance, as the objectives of an optimization problem. It then proposes a compression framework that is able to achieve a more optimal bitrate – task performance trade-off for machines than that of the latest traditional image/video coding method VVC/H.266.

The contributions of this thesis are the followings:

- Proposing a training, evaluation, and inference pipeline that can be trained in an end-to-end fashion using common deep learning algorithms.
- Introducing a novel loss weighting strategy for multi-task training applied to the above pipeline.
- Providing the benchmark results of two learned codecs targeting two different vision tasks: object detection and object segmentation. The results are compared against the performance of the state-of-the-art traditional codec VVC/H.266 on different targeted bitrates.

The rest of this thesis is structured as follows: chapter 2 presents the theoretical background and prior arts in machine-targeting image/video coding. chapter 3 and chapter 4 propose and evaluate an end-to-end pipeline for training image coding for machines codecs. Chapter 5 presents the evaluation results, in comparison to traditional standard VVC/H.266. Further discussion and conclusions are presented in chapter 6.

2 Background

This thesis is proposing a neural network autoencoder based method of image coding in the use cases of Image coding for machines. Additionally, by posing the problem as a multi-task learning problem, the work in this thesis also develops the training strategy based on the prior work in the subject. In the first section of this chapter, the theoretical background of neural networks and image coding is reviewed. Next, the ongoing standardization activities and conducted studies on different paradigms of ICM that are compatible with the traditional image coding standards are presented. Then, the neural network-based studies of image compression and the general concept of autoencoders is discussed. The following section presents the recent techniques of training multi-task neural networks and the last section gives a brief overview of the vision task network architectures that are used in this thesis.

2.1 Preliminary background

2.1.1 Neural networks

In this section, the fundamentals background of the convolutional neural network (CNN) and data compression are reviewed. "Artificial Neural networks" or simply "Neural networks" in this thesis are statistical data models inspired by biological neural networks. In the last 2 decades, because of the long stride ahead in terms of computing infrastructure that is made available mainly via parallel computing frameworks on Graphical Processing Unit (GPU) hardware, NN-based solutions have become more affordable and practical. Neural networks since then have been receiving a lot of attention.

The building block of neural networks is perceptron (Figure 2.1) that receives the signals from other perceptrons in the neural network, processes the input data with its learned weights and bias, and sends the result through a non-linear function, such as sigmoid or ReLU [12], to other perceptrons.

Let $x_1, x_2, ..., x_n \in \boldsymbol{x}$ denote the input signals, the output of the perceptron is defined as:

$$y = f(\sum_{i=1}^{n} w_i x_i + b), w_i \in \theta \tag{2.1}$$

where θ denotes the weights, b denotes the bias of the perceptron, and $f(\cdot)$ is a non-linear function. The weights and biases in literature sometimes are referred to as "parameters" of the neural networks. To simplify the notation, biases are considered

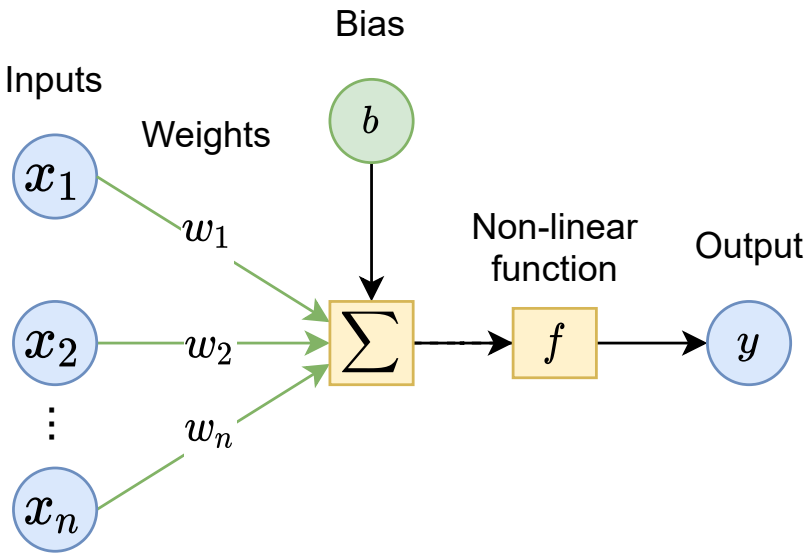


Figure 2.1 *A perceptron in neural networks. The output y is an input to the perceptrons in the next layer.*

as special weights. During the training process of a neural network, its parameters are updated so that the expected distances between the outputs and the targets, i.e., the loss, are minimized. The most common algorithms for updating the parameters are gradient-based methods, in which the weights are updated by a function of the gradients of the loss with respect to the weights:

$$\theta = \theta - \alpha \cdot \nabla_\theta J(\theta), \tag{2.2}$$

where α and $\nabla_\theta J(\theta)$ denotes the learning rate and the gradients with respect to θ, respectively. For more information regarding the neural networks, readers are referred to [13].

Convolutional neural networks

A multi-layer perceptron neural network theoretically can learn to capture any patterns, given it has enough perceptrons. Unfortunately, in such cases, the algorithm complexity grows exponentially. In computer vision, the input data is usually in the form of images or videos that have a large number of individual values for every sub-pixel that would make a multi-layer perceptron infeasible. The convolutional neural network (CNN) is invented to alleviate this problem, inspired by the human visual system. One of the first CNN is introduced in [14], whose authors perform a handwriting recognition task using a neural network with convolutional layers [15] and spatial sub-sampling layers, representing the basic form of CNNs. The convolutional layers convolve a fixed size kernel on their inputs. The outputs of every

layer in a CNN are called "feature maps", which are treated as inputs by the following layers. The spatial sub-sampling layers in between the convolution layers are to widen the receptive field of the neural network so that it could capture spatial features in different sizes.

A CNN, by mimicking the "receptive field" of human vision, can effectively capture the visual patterns on the input image with much fewer parameters because the parameters of fully-connected layers are now replaced by the shared parameters of the kernels. Fewer parameters means faster and smaller model hence less dependent on the hardware devices. Additionally, the linear convolution operator can be implemented as matrix multiplications and run in parallel by GPUs. Besides, CNNs inherit a few merits from the human vision that make them suitable for vision tasks: the trained parameters in CNN tend to be robust to visual features of different scales, positions, or rotations.

2.1.2 Entropy coding

The common paradigm of image/video coding in many coding standards, e.g. JPEG, JPEG-2000 or VVC/H.266 [2, 3, 5, 6] is *transform coding* [16], where the input data is transformed into a more compressible representation, such as DCT transforms in JPEG, VVC or Discrete Wavelet Transform in JPEG-2000. The compressibility is measured by the number of bits needed to encode the data into a bitstream. This number is lower-bounded by Shannon entropy [17], thus this bitstream coding step in image/video coding is also known as "Entropy coding". There are two widely adopted entropy coding families of algorithms that can almost guarantee the lower-bound is achieved, given a long enough string of input symbols: arithmetic coding and asymmetric numeral systems (ANS).

In arithmetic coding, a string of symbols will be encoded into one float number. The probability of the next symbol determines which sub-range of value that symbol will be encoded into. For example, given a dictionary of 4 symbols A, B, C, D with the probabilities in Table 2.1. The string of symbols "ACD" would be encoded as shown in Figure 2.2.

Symbol	Probability
A	0.6
B	0.2
C	0.1
D	0.1

Table 2.1 *Probability distribution*

The arithmetic encoder and decoder need to look upon the Table 2.1 to accomplish their tasks, therefore this table is also referred to as "prior distribution".

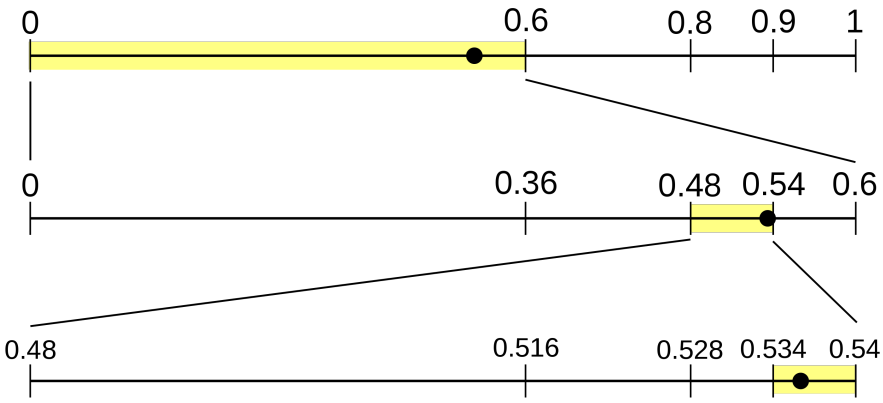


Figure 2.2 *Encoding of symbol string "ACD" . Source: [18] (Public domain).*

In the case of "adaptive" arithmetic coding, this table would change at every step according to the "context". As for asymmetric numeral systems, instead of looking into sub-ranges of the current range, this algorithm to scales up the range by a factor based on the prior distribution. ANS may not be as optimal in terms of compression efficiency as arithmetic coding, but in return almost as fast as Huffman coding. This graceful trade-off makes ANS a useful family of entropy coding algorithms in practice.

The entropy model (sometimes referred to as prior probability model) tries to predict this table for the entropy coder. The more accurate it is, the lower entropy it gets, lower-bounded by the entropy of the unknown actual distribution of the symbols.

2.2 Traditional standards-based approaches to image coding targeting machine consumption

2.2.1 Standardization activities

The conventional way of using traditional codecs for task networks exposes its weakness in the explosion of IoT (Internet of Things) devices when the amount of visual signal data that is the material of machine-to-machine communications exponentially growing. Here arise the concerns whether the traditional codecs for visual signal compression are efficient enough for this new era. In order to address this, Moving Picture Experts Group (MPEG) has issued the standardizations of Compact Descriptors for Visual Search (CDVS) [19], Compact Descriptors for Visual Analysis (CDVA)[20] and lately, Video Coding for Machines (VCM) Ad-hoc group [21]. CDVS and CDVA explore the "Analyze then Compress" paradigm, as opposed

to "Compress then Analyze", where the input data is compressed and transmitted to the machines for visual features extraction and analysis. In "Analyze then Compress", instead of transmitting the visual signal data, its visual features are extracted and transmitted instead. Here are a few key points of these standardization activities:

CDVS aims to improve the visual searching speed and accuracy in mobile devices. It specifies a normative feature extraction process with hand-crafted descriptors. This approach also helps distribute the work of feature descriptors extraction. However, it has difficulties handling more complicated analysis, where the hand-crafted descriptors of the images are not sufficient for good task performance.

CDVA further improves the task performance by adopting the feature descriptors based on deep neural networks into a normative feature extractor. This approach is being explored, initial studies have shown promising results [22]. In [23], the authors use the CDVA descriptors as the input for autonomous agents of different types to make decisions or detection actions. The current limitation of this approach is, however, it needs a generic deep neural network that can extract features for broad use. Given the fast-changing development of the deep learning community, finding such NNs is challenging.

VCM Ad-hoc group explores the techniques of compressing video primarily for machine-consumption, and optionally also for human analysis. For example, a surveillance camera sends its signals directly to the task network for anomaly detection with no human interaction, but when a human agent wants to investigate the footage, there might be a way to reconstruct the human-consumable signal with an additional bitstream of data. The requirement documents for this study group are still in an early stage, but it is expected to bridge the gap between traditional video coding and other machine-targeting video coding techniques. At the time of this thesis, the VCM group was focusing on a few major computer vision tasks, namely: object detection, object tracking, instance segmentation, action recognition and event detection. A more complete list of the key tasks can be found in [24]. The work of this thesis falls within the scope of VCM activities.

2.2.2 Feature maps compression

In a computer vision system, the task networks are located on the receiver-side, thus the devices on this side are required to have enough computational power for the task. When dealing with a centralized system, for instance, a smart city IoT system in which the cloud center receives data from the edge devices and returns

the vision task results such as object detection back to them. The bandwidth and computational requirement for this central unit could be prohibitively expensive when the number of edge devices grows. By following the "Analyze then Compress" strategy, authors of [25] and [26] alleviate this problem by splitting the task network and putting the parts into both sides, off-loading the computational demand on the cloud center. The sender-side devices extract and pack the intermediate feature maps at the splitting point, compress them using a standard video codec (HEVC/H.265) and send the bitstream to the other side. The compressed features are unpacked on the receiver-side and then are fed into the other part of the task network for task results. The intermediate feature maps of a vision task network usually have smaller spatial sizes and more channels than those of the natural input image, e.g. 512 channels. Therefore to compress these feature maps using a standard video codec that is designed for 3-channel images, the feature maps tensor would be transformed into a compatible format. Authors of [25, 26] experiment with tiling the feature maps for a big grayscale image or stacking them along the temporal axis with a determined order of correlation level. The transformed tensor would be treated as a normal video by the video codec, with their pseudo-spatiotemporal redundancy analyzed and exploited for compression. So far none of the work in this direction has reported coding efficiency in comparison with the conventional scheme. Additionally, the application of feature compression-based methods to improve the coding efficiency encounters the following challenges:

- The traditional codecs are designed for natural image formats (e.g., 3-channel color image), while the feature maps tend to have arbitrary dimensional sizes or even be in arbitrary dimensionality. In order to deal with this issue, one can reshape the feature maps to satisfy the input format requirement, such as proposed in [25, 26]. However, this is clearly an unexpected behavior to the traditional image codecs, and the algorithms determining the order of the feature maps in the new pseudo-image are still open for study.

- Moreover, in many cases, the intermediate feature maps of a neural network contain much more latent elements than the number of sub-pixels in the input image. On top of that, these latent elements have different value distributions than that of natural images, for which the traditional codecs are optimized. Therefore the coding efficiency of a traditional codec in this case is likely to be sub-optimal.

- Finding a clean "splitting point" can be challenging as the model architectures in deep learning have been adopting more complicated structures, for example, the residual connections [27] or feature pyramid architecture [28] may limit the options for such points.

2.2.3 Other standard-compliant approaches

To directly optimize the video codecs for task performance, some authors endeavor to modify the video codecs that are designed for humans, while preserving the coded bitstream format, thus keeping it still compatible with the unmodified modules.

In [29], the authors first compare the task performance on HEVC/H.265 coded data with on VVC/H.266 coded data. The bitrate saving gain with respect to task performance when using the new VVC codec is not as much as the saving gain concerning the human vision metric. Furthermore, they found that almost all of the in-loop filters have a negative impact on the bitrate saving and computational complexity for the machine-targeting pipeline. By disabling these filters, a significant bitrate saving gain can be achieved. Again this work shows that the traditional codecs are not optimal for computer vision tasks.

The work in [30] proposes to reformulate the conventional rate-distortion optimization (RDO) in VVC/H.266 by replacing pixel-based distortion metrics with feature-based distortion metrics, consequently guides the standard encoder to reduce the distortion of the features for the task network, thus improves the coding efficiency. The authors of [31] on the other hand fine-tune the weights of JPEG XS [32] "High" profile for task performance.

Although the above methods manage to improve coding efficiency or reduce the computational complexity, they don't aim to completely replace the conventional pipeline for human-oriented coding but instead, they only add incremental capabilities to the system.

2.3 Neural network based image codecs

Although Image coding for machines is still a new research field, the use of neural network-based modules to aid or completely replace the traditional image codec has already been actively studied recently.

In [33], a CNN post-filter is used to further enhance the image quality, while [34–37] seek to effectively model the distribution of the input images that is provided to the entropy coder for lossless compression. Additionally, auto-encoders have been widely used as the neural network-based image codecs in recent work for end-to-end training. More details regarding the distribution modeling and auto-encoder codecs will be discussed in the next subsections.

2.3.1 Prior probability distribution modeling

Autoregressive context models - Pixel RNN/CNN and Pixel CNN++

The bold idea of this scheme is first proposed in [34] with Pixel Recurrent Neural Network (Pixel RNN) and Pixel Convolutional Neural Network (Pixel CNN). The main goal of Pixel RNN/CNN is to model the prior distribution of the input image $\boldsymbol{x}$ with an autoregressive model illustrated in Figure 2.3. In an autoregressive model, a value x_n is conditioned on the previous values $x_{n-1}, x_{n-2}, ..., x_{n-i}$. In the proposed scheme, the dependencies are in the sub-pixel level, i.e. x_n denotes the channel-wise value. With a powerful RNN, in Pixel RNN, i is unbounded and the dependencies can go all the way to the start, i.e. x_1. However, this creates a burden of computational complexity, therefore the authors also adapt their idea to a CNN architecture (Pixel CNN), in which the dependencies are bounded naturally by the receptive fields of the masked convolutional kernels.

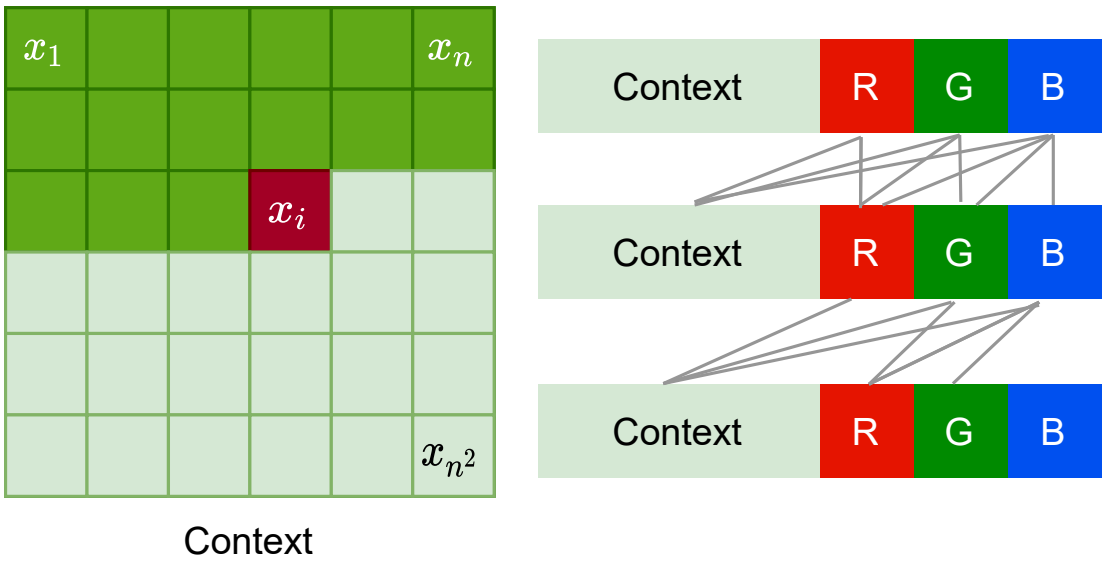


Figure 2.3 *The autoregressive model in [34].* ***Left:*** *Pixel x_i is conditioned on all previously generated pixels (context).* ***Right:*** *The connectivity of the masked convolutions. In the first layer, each channel connects to the context and the previous channels. In the subsequent layers, the channels are also connected to themselves.*

While Pixel RNN/CNN demonstrates superior compression efficiency, its encoding speed and computational power requirements are impractical for broad usage. To this end, [35] further improves the coding speed by using a discretized logistic mixture likelihood to model the element values, hence reduce the number of parameters, together with relaxing the sub-pixel level to whole-pixel level dependencies, and other modifications. Even though the speed improvement is significant, it still cannot take advantage of parallel computational power in GPUs for feasible runtime

because of the autoregressive "context" nature of this scheme in general.

Lossless image compression through super-resolution - SRec

The authors of [37] proposed a hierarchical learning scheme that involves distribution estimation at multiple scales. The overview of this model is shown in Figure 2.4. The goal of this model is to learn the distribution $P(x^l|x^{l+1})$, where x^{l+1} denotes the downsampled version of x^l, this step can be done multiple times at multiple scales. As a result, only the downsampled image are encoded, together with the super-resolution distributions for every scale with the expectation that the total size of the coded bitstreams is smaller than that of the full-size image. The downsampling is done by Average Pooling with a stride of 2, meaning the average of 4 pixels in x^l produce 1 pixel in the downsampled image x^{l+1}. This deterministic behavior makes the fourth pixel in the group can be obtained for free given the other three during the upsampling process, consequently further increases the bitrate saving gain.

Figure 2.4 *Image compression with super-resolution pipeline in [37]. Instead of compressing the full-size image, this scheme compresses the smallest-scale image and the lossless super-resolution bitstreams (SR bitstreams) of the subsequent scales.*

Lossless image compression using multi-scale feature maps – L3C

In L3C [36], the probability distribution model is a factorization of the component distributions learned at different scales of the feature pyramid. The feature elements are modeled by discretized logistic mixtures that are conditioned on the prior from the previous scales as shown in Figure 2.5. By using non-autoregressive models at almost every step, encoding and decoding using this architecture are orders of magnitude faster than Pixel CNN. This makes L3C a practical model, although its compression efficiency is not as good as Pixel CNN.

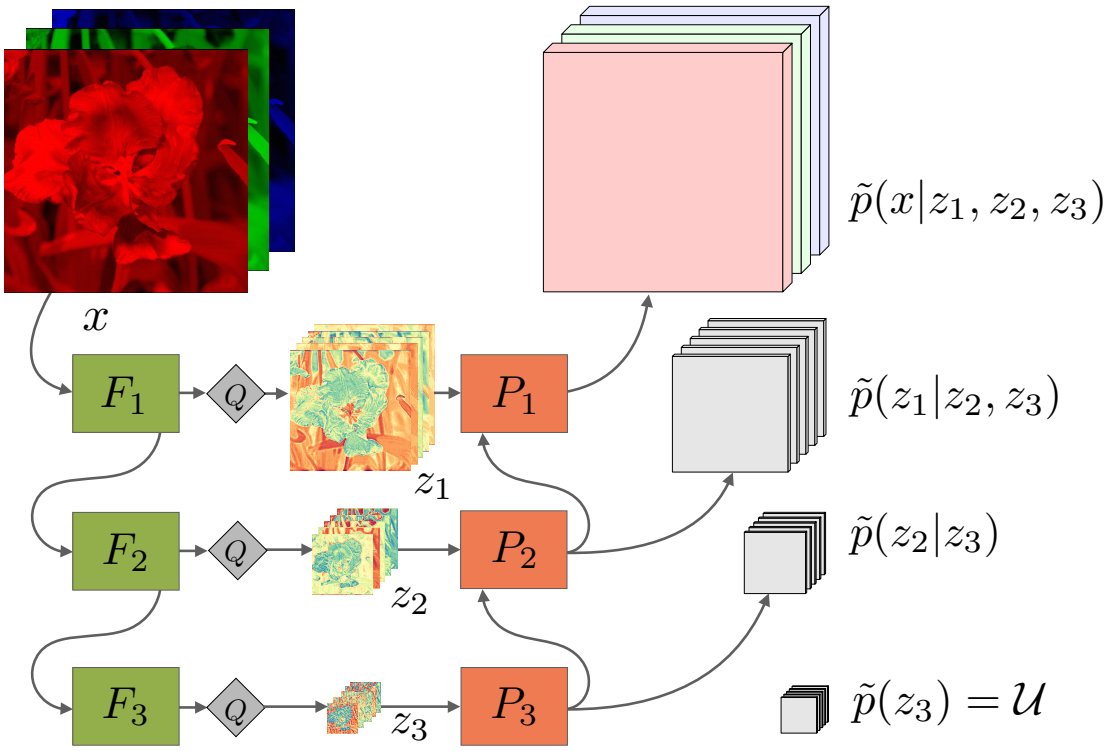


Figure 2.5 *(Courtesy of the authors of [36], permissions to use this figure and its caption are given for this thesis): Overview of the L3C model architecture. At every scale $s \in [1, S]$, the output of the feature extractor F_s is quantized as z_s. The joint distribution $\tilde{p}(x, z_1, ..., z_S)$ is modeled using the non-autoregressive predictors P_s.*

Hyper-prior models

In [38–40], the authors propose end-to-end pipelines for image *transform coding* [16], where the image's latent representation and its distribution are entropy encoded instead using hierarchical learned *hyperprior*. This is a line of work that is utilized in the proposed method of this thesis, in which the main targeted end-users are the task networks, and the main coder is substituted by a more capable one. The details of these models are discussed later in section 3.2 of chapter 3.

2.3.2 Auto-encoder in image coding

An auto-encoder in the scope of this thesis is a neural network architecture that consists of an encoder and a decoder. Figure 2.6 illustrates an overview of autoencoders. Auto-encoders have been widely adopted to learn the semantic features or to reduce the dimensionality of the input. Many proposals on NN-based image codecs are using this structures, e.g. [36, 38–40]. Given an input $\boldsymbol{x}$, the encoder $\boldsymbol{E}(\cdot)$ takes $\boldsymbol{x}$ as input and transforms it to a latent representation $\boldsymbol{z}$. The decoder $\boldsymbol{D}(\cdot)$ then

inverses the latent representation $\boldsymbol{z}$ to reconstruct the input as $\hat{\boldsymbol{x}}$. Formally:

$$\begin{aligned} \boldsymbol{z} &= \boldsymbol{E}(\boldsymbol{x}; \boldsymbol{\theta_E}) \\ \hat{\boldsymbol{x}} &= \boldsymbol{D}(\boldsymbol{z}; \boldsymbol{\theta_D}), \end{aligned} \tag{2.3}$$

where $\boldsymbol{\theta_E}$ and $\boldsymbol{\theta_D}$ denote the parameters of the encoder and decoder, respectively. The approximation relationship between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ is usually imposed by minimizing the difference between them, for example using a ℓ_2-norm distance:

$$d = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2 \tag{2.4}$$

In image coding, the difference between $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ is also referred to as "distortion". In order to avoid trivial solutions, for example, both $\boldsymbol{E}(\cdot)$ and $\boldsymbol{D}(\cdot)$ are identity functions, in many cases, the latent dimensionality is forced to be narrower than that of the input, hence the information capacity is also smaller. This way, the parameters $\boldsymbol{\theta_E}$ of $\boldsymbol{E}(\cdot; \boldsymbol{\theta_E})$ have be adjusted so that information contained in the latent representation $\boldsymbol{z}$ is the most crucial. Consequently, the parameters $\boldsymbol{\theta_D}$ of $\boldsymbol{D}(\cdot; \boldsymbol{\theta_D})$ have to adapt to the latent representation of the input image in order to reconstruct it. As a result, the encoder learns to present the input in a more compact way (compress), and the decoder learns to reconstruct the original input from the encoded tensor (decompress).

In image coding, the "compact" nature is measured by bits per pixel (bpp or BPP), meaning the average number of bits that are used to store the information for each pixel on the original image. This metric is also referred to as "bitrate" or "rate" interchangeably in this thesis.

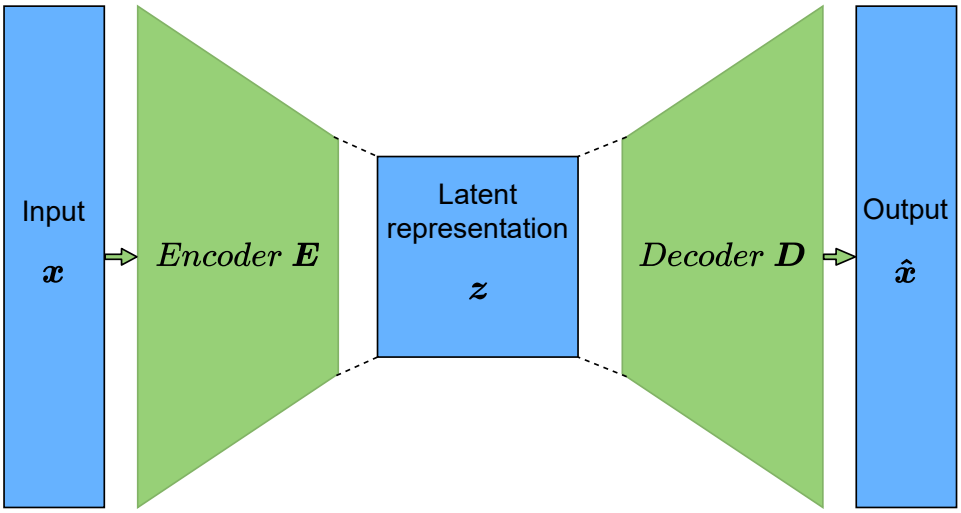


Figure 2.6 *Basic architecture of an autoencoder.*

The use of residual connections in auto-encoders

The concept of residual connections is first introduced in [27]. It is proved that residual connections can help neural networks go much deeper without gradient vanishing or gradient exploding. The use of residual connections is also studied and experimented on auto-encoders. In [41] shown the importance of residual connections for better representation learning. They significantly boost the performance on both classification and reconstruction tasks, in comparison with a system without residual connections. The authors of [42] study auto-encoders usage with residual connections in text document image restoration, in which the combination of two types of residual connections yields superior performance to the other compared methods.

2.4 Neural network training techniques for multi-task system

Conducted study in [43–46] give evidences of the complicated trade-off relationship between vision task performance, rate, perception and distortion in image coding. The authors of [43, 44] prove the existence of the three-way trade-offs of classification error rate, distortion, and the perceptual differences that cannot be optimal at the same time. Similarly, [46] shows that bitrate, distortion, and the perceptual differences are at odds with each other. In an end-to-end system such as the one being proposed in this thesis, the goal is to find the optimal trade-offs between the objectives, i.e. bitrate and task prediction error. Thus, balancing the objectives is a critical matter.

On the other hand, in a multi-task neural network system, the training loss is a combination of the component losses corresponding to the objectives. A naïve training loss is given by a weighted summation of the component losses:

$$\mathcal{L}_{total} = \sum_{i=1}^{N} w_i \cdot \mathcal{L}_i, \tag{2.5}$$

where N denotes the number of component losses and w_i denotes the weight for component loss $\mathcal{L}_i$. The authors of [47] provide a multi-task training overview, which discusses the importance of loss weighting and gives highlights to a few automatic loss balancing techniques, as opposed to fixed weighted losses. These techniques are summarized below:

Dynamic weight average [48]: The authors via this work propose a dynamic loss weighting strategy that considers the rate of loss change of each task at each

iteration. The weight of task i at iteration t is given by:

$$\begin{aligned} w_i(t) &= N \cdot \frac{\exp(\lambda_k(t-1)/T)}{\sum_{a=1}^{N} \exp(\lambda_a(t-1)/T)}, \\ \text{with } \lambda_i(t-1) &= \frac{\mathcal{L}_i(t-1)}{\mathcal{L}_i(t-2)}, \end{aligned} \tag{2.6}$$

In this equation, λ_i expresses the changing rate of $\mathcal{L}_i$ and T controls the "softness" of the weighting: a large T results in more even distribution between tasks.

Loss weighting using uncertainty [49]: This approach balances the losses by maximizing the likelihood with task-dependent uncertainty. For instance, let $\boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F})$ be the output of a NN-based regression model with parameters $\boldsymbol{\theta_F}$ given the input $\boldsymbol{x}$. We can define the likelihood of this model as a Gaussian as follow:

$$p(\boldsymbol{y}_i|\boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F})) = \mathcal{N}(\boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F}), \sigma_i^2), \tag{2.7}$$

where $\boldsymbol{y}_i$ denotes the target of task i and σ_i denotes the observation noise. For multiple tasks, the likelihood becomes a factorization over the outputs:

$$p(\boldsymbol{y}_1, ...\boldsymbol{y}_N|\boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F})) = p(\boldsymbol{y}_1|\boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F}))...p(\boldsymbol{y}_N|\boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F})) \tag{2.8}$$

Then one can design their loss function to maximize the likelihood with respect to σ_i to balance the losses. In practice, the loss function weighted by this method is often given by:

$$\mathcal{L}_{total} = \sum_{i=1}^{N} \left(\frac{1}{2\sigma_i^2} \cdot \mathcal{L}_i(\boldsymbol{y}_i, \boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F})) + ln(\sigma_i^2) \right), \tag{2.9}$$

where σ_i are trainable parameters.

Loss weighting using uncertainty with positive regularization term [50]: The loss function in Equation 2.9 can yield negative value when $\sigma_i^2 < 0$. This work adapt the loss weight technique in [49] by simply modify the regularization term from $ln(\sigma_i^2)$ to $ln(\sigma_i^2 + 1)$, enforcing positive value for the losses. The revised equation is given by:

$$\mathcal{L}_{total} = \sum_{i=1}^{N} \left(\frac{1}{2\sigma_i^2} \cdot \mathcal{L}_i(\boldsymbol{y}_i, \boldsymbol{F}(\boldsymbol{x}; \boldsymbol{\theta_F})) + ln(\sigma_i^2 + 1) \right) \tag{2.10}$$

These techniques automatically balance the loss weights based on their statistical analysis. While this is a nice feature, they don't directly offer full control over the

priorities of the objectives. In the work of this thesis, it is a key feature for successful codec training. Therefore a new non-fixed (dynamic) loss weighting strategy is presented in subsection 3.4.1.

2.5 Neural network for computer vision tasks

The neural network architectures that are used for object detection and instance segmentation in this thesis will be reviewed in this section.

2.5.1 Object detection

This vision task seeks to identify the rectangular bounding box of the object instances in the input image. The network architecture for this task is Faster R-CNN [51], which aims to improve the region proposals in Fast R-CNN [52]. Fast R-CNN is an object detection network that detects and classifies objects on each region of interest (RoI) proposal. The region proposals in Fast R-CNN are found by a selective search algorithm [53]. In essence, the training loss of this network is made up of two components: object classification loss and bounding-box regression loss, denoted by $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$, respectively. The overview of Fast R-CNN is shown in Figure 2.7.

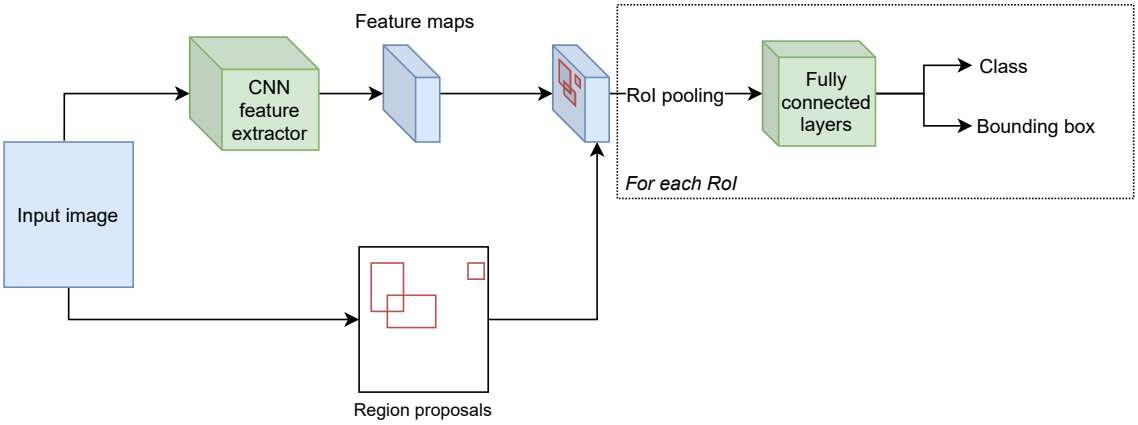


Figure 2.7 *Fast R-CNN architecture for object detection as described in [52]. The regional proposals are given by a selective search algorithm. RoI pooling extracts the features corresponding to the region proposals and feeds them to the classifier.*

In Faster R-CNN, instead of using a selective search algorithm, the authors introduce a Region Proposals Network (RPN) in order to boost the detection speed of the system by sharing the feature extraction layers with the detection branch, which is essentially the Fast R-CNN network. The architecture of Faster R-CNN is given by Figure 2.8. Outputs of the RPN network contains two parts: "objectness" indicates whether the proposed region is an object or background, and bounding box regression for the offsets from the pre-defined anchors. Therefore the training

loss for this module has 2 components: objectness loss $\mathcal{L}_{obj}$ and region proposal regression loss $\mathcal{L}_{preg}$. Formally, the two branches of RPN and object classifier are optimized with a training loss given by:

$$\mathcal{L}_{task} = \underbrace{\mathcal{L}_{cls} + \mathcal{L}_{reg}}_{\text{classifier branch}} + \underbrace{\mathcal{L}_{preg} + \mathcal{L}_{obj}}_{\text{RPN branch}} \tag{2.11}$$

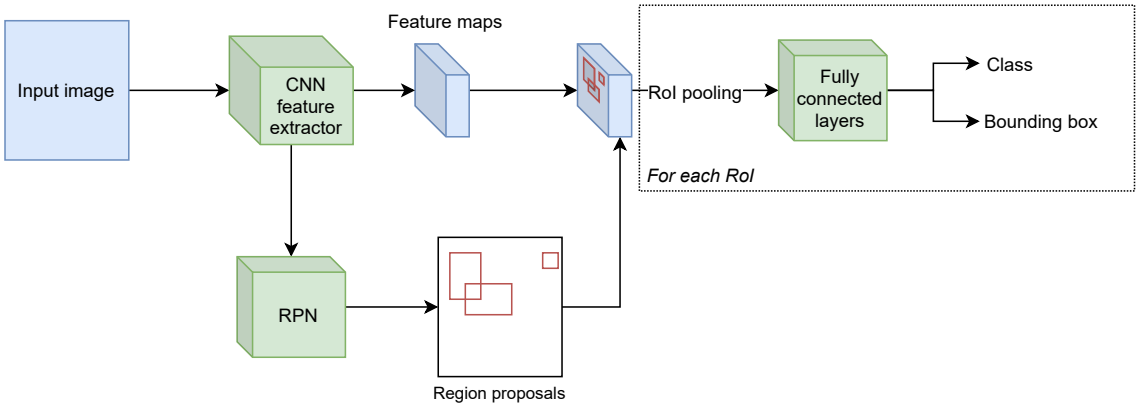


Figure 2.8 *Faster R-CNN architecture for object detection as described in [51]. The region proposals are suggested by a region proposal network (RPN).*

2.5.2 Instance segmentation

The Mask R-CNN framework [54] shown in Figure 2.9 is used for task instance segmentation. The goal of instance segmentation is to detect the object boundaries at pixel-level for every object instance. This network architecture is simply an

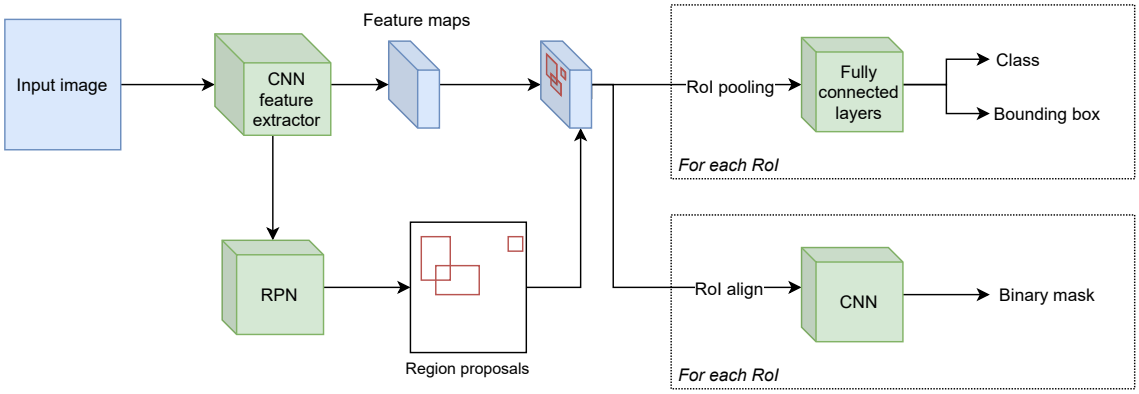


Figure 2.9 *Mask R-CNN architecture for instance segmentation as described in [54] Another branch dedicated to binary mask prediction is added to the system.*

extension of the Faster R-CNN discussed in the previous subsection. It adds a segmentation mask prediction branch to predict the masks on each RoI in parallel with

the existing Fast R-CNN branch for object classification. The new mask prediction branch is independent of the classifier branch, since it predicts the binary masks for every class, regardless of the object classification. This branch is optimized by imposing the minimization of segmentation loss, denoted by $\mathcal{L}_{mask}$ The training loss for this network is given by:

$$\mathcal{L}_{task} = \underbrace{\mathcal{L}_{cls} + \mathcal{L}_{reg}}_{\text{classifier branch}} + \underbrace{\mathcal{L}_{preg} + \mathcal{L}_{obj}}_{\text{RPN branch}} + \underbrace{\mathcal{L}_{mask}}_{\text{mask branch}} \tag{2.12}$$

3 Proposed method

The proposed Image Coding for Machines (ICM) system has three main neural components which are shown in Figure 3.1: a pair of encoder-decoder (autoencoder) as the main coder, an entropy model, and a task network. In contrast to the end-to-end framework in [40], the proposed system targets task performance instead of pixel-domain fidelity. When one needs to perform a specific task on an image $\boldsymbol{x}$, the image is sent to the encoder network $\boldsymbol{E}(\cdot;\boldsymbol{\theta_E})$ to produce a more compressible latent representation $\boldsymbol{y} = \boldsymbol{E}(\boldsymbol{x};\boldsymbol{\theta_E})$, which is then quantized and losslessly compressed by the Entropy Encoder (EE) using the probability distribution $p_{\hat{\boldsymbol{y}}}$ estimated by the Entropy model (discussed in section 3.2). The output bitstream of this compression is transmitted to the receiver-side to be decompressed and eventually reconstructed by the decoder $\hat{\boldsymbol{x}} = \boldsymbol{D}(\hat{\boldsymbol{y}};\boldsymbol{\theta_D})$. The task network analyzes $\hat{\boldsymbol{x}}$ as input and returns the task results.

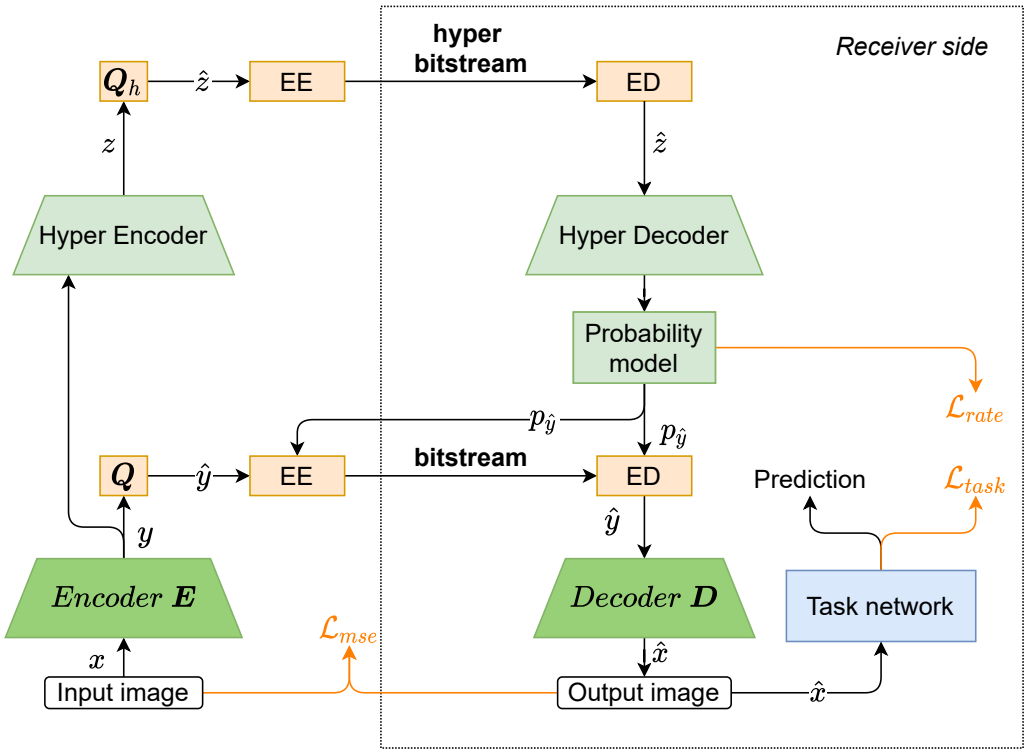
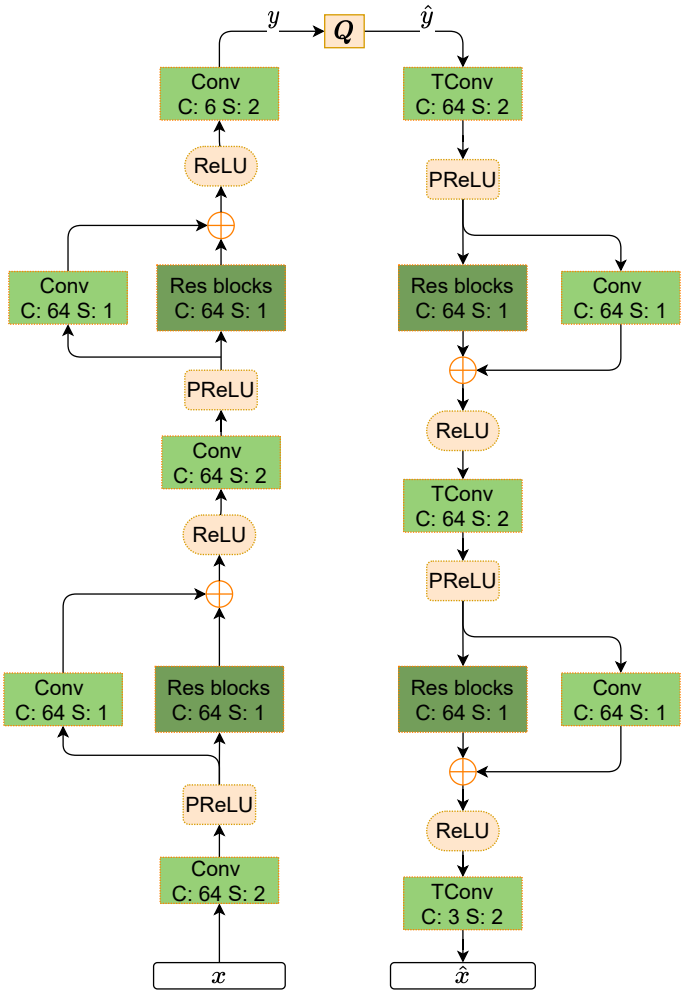


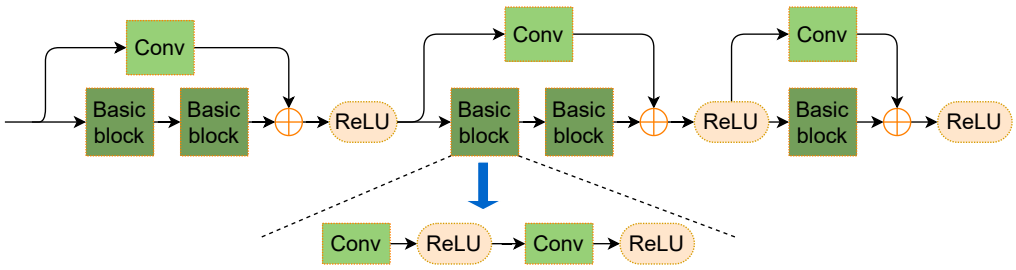
Figure 3.1 *Overview of the proposed ICM system. The main codec consists of the neural network Encoder* $\boldsymbol{E}$ *and Decoder* $\boldsymbol{D}$*. The Hyper Encoder and Hyper Decoder learn the hyperprior from the latent tensor* $\boldsymbol{y}$*. The Probability model estimates the probability distribution using the hyperprior and provide it for the entropy encoder (EE) and entropy decoder (ED). The task network in this system is a frozen pretrained network and provide task loss* $\mathcal{L}_{task}$ *during the training stage and prediction in the evaluation stage.*

3.1 Auto-encoder

Instead of reducing distortion in the reconstructed signal like other common auto-encoders, the proposed auto-encoder aims to produce a data tensor of the same



(a) Auto-encoder overview



(b) Inside the "Res blocks" blocks

Figure 3.2 *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.*

size as the input image that provides good task performance while consuming a low bitrate. The architecture for the autoencoder is built from blocks of convolutional layers and skip connections as shown in Figure 3.2. For non-linearity, ReLU [12] and PReLU [55] are applied on output of each convolutional block or layer. Instead of reducing distortion in the reconstructed signal like other common auto-encoders, the proposed auto-encoder aims to produce a data tensor of the same size as the input image that provides good task performance while consuming a low bitrate. These two objectives are formally denoted as task loss $\mathcal{L}_{task}$ and rate loss $\mathcal{L}_{rate}$ respectively.

The number of channels of the intermediate layers and the last layer are considerably low to remain a compact model size for a better encoding speed and low resource consumption. This auto-encoder is optimized with the aforementioned loss terms. The details for this training will be discussed in section 3.4

Let $\boldsymbol{x}, \hat{\boldsymbol{x}} \in \mathbb{R}^{W \times H \times 3}$ denote a pair of color images of spatial size $W \times H$. The encoder transforms the input $\boldsymbol{x}$ into a latent representation $\boldsymbol{y} = \boldsymbol{E}(\boldsymbol{x}; \boldsymbol{\theta_E})$, with $\boldsymbol{y} \in \mathbb{R}^{\frac{W}{8} \times \frac{H}{8} \times 6}$ and $\boldsymbol{\theta_E}$ denotes the learnable parameters of the encoder. The latent tensor $\boldsymbol{y}$ is then quantized by the quantizer $\hat{\boldsymbol{y}} = \boldsymbol{Q}(\boldsymbol{y})$ and sent to the decoder for a reconstruction: $\hat{\boldsymbol{x}} = \boldsymbol{D}(\hat{\boldsymbol{y}}; \boldsymbol{\theta_D})$. During training process the quantization step is relaxed to make gradients back-propagation possible, the details are discussed in subsection 3.4.2.

3.2 Entropy model

For entropy coding, the proposed pipeline uses an asymmetric numeral systems (ANS) [56] codec. An ANS entropy encoder first encodes a stream of symbols into a single natural number according to the probability distribution of the symbols then converts the number into a binary bitstream. The proposed ICM system aims to encode the quantized latent $\hat{\boldsymbol{y}}$ with the unknown associated marginal distribution $m_{\hat{\boldsymbol{y}}}$ into a bitstream with minimum code length, which is lower-bounded by the Shannon entropy [17]. Since $m_{\hat{\boldsymbol{y}}}$ arises from both the unknown input image distribution $p_{\boldsymbol{x}}$ and the transformation method $\boldsymbol{Q}(\boldsymbol{E}(\cdot; \boldsymbol{\theta_E}))$, the code length can only be estimated by Shannon cross-entropy:

$$\begin{aligned} R &= \mathbb{E}_{\hat{\boldsymbol{y}} \sim m_{\hat{\boldsymbol{y}}}} \left[-\log_2 p_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}}) \right] \\ &= \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} \left[-\log_2 p_{\hat{\boldsymbol{y}}}(\boldsymbol{Q}(\boldsymbol{E}(\boldsymbol{x}; \boldsymbol{\theta_E}))) \right], \end{aligned} \tag{3.1}$$

where $p_{\hat{\boldsymbol{y}}}$ denotes the estimated distribution of $\hat{\boldsymbol{y}}$. The Shannon cross-entropy is minimized when $p_{\hat{\boldsymbol{y}}}$ is identical to $m_{\hat{\boldsymbol{y}}}$. The entropy model seeks to learn $p_{\hat{\boldsymbol{y}}}$, also known as "prior" of $\hat{\boldsymbol{y}}$ in order to minimize R. The next sub-sections present the entropy model used in this ICM system.

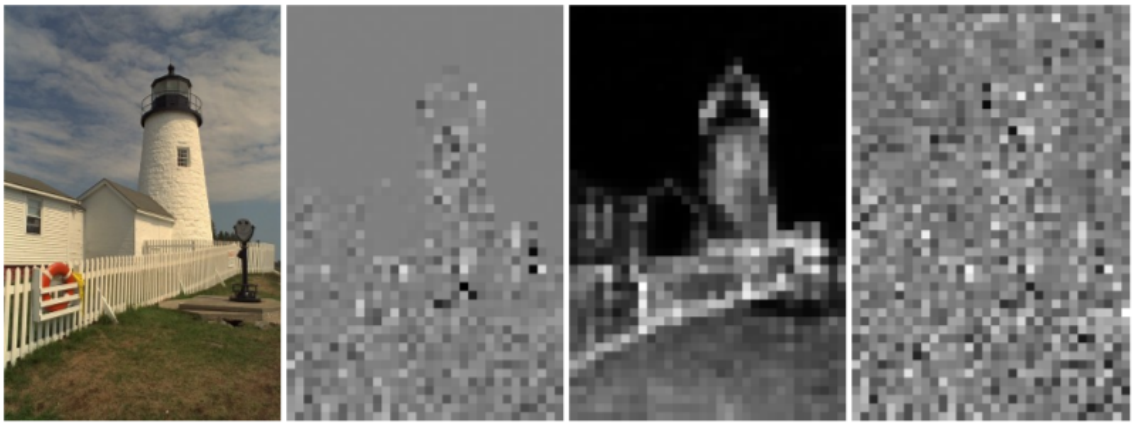


Figure 3.3 *(Courtesy of the authors of [39], permissions to use this figure and its caption are given for this thesis):* ***Left:*** *an image from the Kodak dataset [57].* ***Middle left:*** *visualization of a subset of the latent representation* $\boldsymbol{y}$ *of that image, learned by a factorized-prior model. Note that there is clearly visible structure around edges and textured regions, indicating that a dependency structure exists in the marginal which is not represented in the factorized prior.* ***Middle right:*** *standard deviations* $\hat{\boldsymbol{\sigma}}$ *of the latents as predicted by the model augmented with a hyperprior.* ***Right:*** *latents* $\boldsymbol{y}$ *divided element-wise by their standard deviation. Note how this reduces the apparent structure, indicating that the structure is captured by the new prior.*

3.2.1 The scale-hyperprior model

This model is originally proposed in [39] based on the intuition that the spatially neighboring elements of the latent representation $\boldsymbol{y}$ tend to vary together in their scales (standard deviations). This structure dependency is usually not well-captured by a factorized-prior model, resulting in sub-optimal compression efficiency. Therefore the authors propose a "hyperprior" architecture in order to capture this dependency. Their example illustration of the spatial scale-variation is shown in Figure 3.3.

The authors of [39] model the elements of the approximated quantized latent tensor $\tilde{\boldsymbol{y}}$ by a zero-mean Gaussian:

$$\tilde{y}_i = \mathcal{N}(0, \sigma_i^2) + \mathcal{U}(-\frac{1}{2}, \frac{1}{2}), \tag{3.2}$$

where standard deviation ("scale") σ_i is predicted by the hyperprior model, and the uniform distribution $\mathcal{U}(\cdot)$ is the approximation of quantization step (further discussed in subsection 3.4.2) The probability of $\hat{y}_i$ is then given by a closed-form formula:

$$p_{\hat{\boldsymbol{y}}}\left(\hat{y}_i \mid \hat{\sigma}_i\right) = p_{\tilde{\boldsymbol{y}}}\left(\hat{y}_i \mid \hat{\sigma}_i\right) = \left(\mathcal{N}\left(0, \hat{\sigma}_i\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)\left(\hat{y}_i\right) = \int_{\hat{y}_i - 1/2}^{\hat{y}_i + 1/2} \mathcal{N}\left(y \mid 0, \hat{\sigma}_i\right) \mathrm{d}y \tag{3.3}$$

Because of this formulation, $\hat{\boldsymbol{\sigma}}$ is considered as the "prior" of $p_{\hat{\boldsymbol{y}}}$, making it the "hyperprior" of $\hat{\boldsymbol{y}}$.

The "hyperprior model" in general includes another auto-encoder structure that takes the latent $\boldsymbol{y}$ as input, stacked on top of the main auto-encoder. The "hyperprior" tensor $\hat{\boldsymbol{\sigma}}|\tilde{\boldsymbol{\sigma}}$ has the same dimensionality as that of the quantized latent tensor $\hat{\boldsymbol{y}}|\tilde{\boldsymbol{y}}$, meaning each element $\hat{y}_i|\tilde{y}_i$ of $\hat{\boldsymbol{y}}|\tilde{\boldsymbol{y}}$ will get one scale prediction $\hat{\sigma}_i|\tilde{\sigma}_i$. The pipeline overview is shown in Figure 3.4.

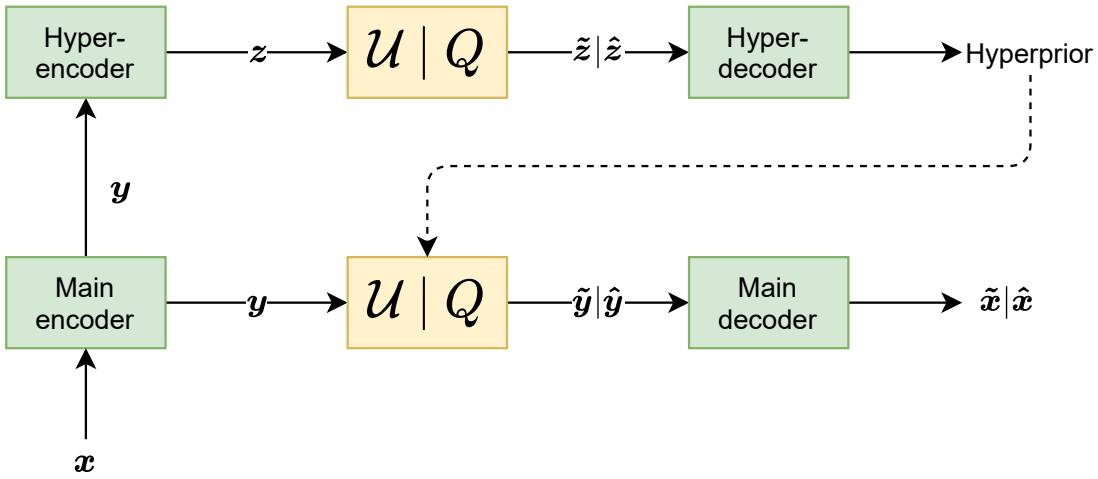


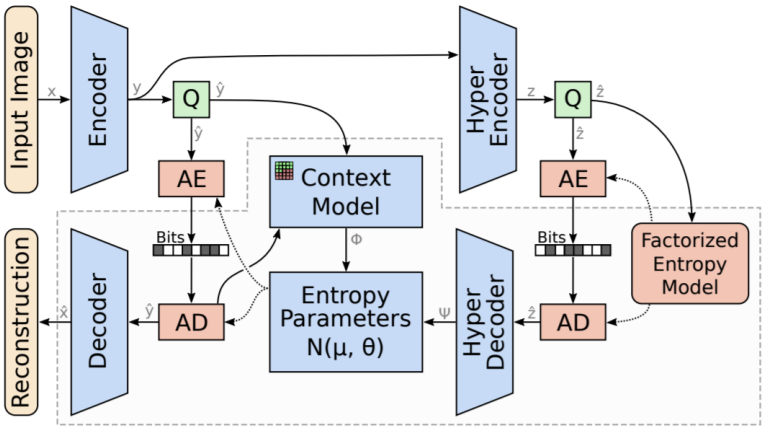
Figure 3.4 *General overview of the "hyperprior" approaches. The hyper encoder and hyper decoder pipeline are stacked on top of the main coder, take the latent $\boldsymbol{y}$ as its input. The "hyper-latents" is denoted as $\boldsymbol{z}$. The hyperprior learned from the hyper coder pipeline is use for distribution estimation of $\boldsymbol{y}$.*

3.2.2 The joint autoregressive and hierarchical priors model

Extending the "scale hyperprior" model [39] (subsection 3.2.1), this model, proposed in [40], captures more correlations in the latent representation $\boldsymbol{y}$ by:

- Switching from zero-mean Gaussian scale models to Gaussian mixture models for the latent elements modeling. That means the hyperprior will also include the means of the Gaussians, which are assumed to be zero-mean in [39]. The output hyperprior tensor therefore has double the number of channels for both means and scales.

- Adding an autoregressive context model and another NN model to combine the context information with the output of the hyper-autoencoder in order to produce the final hyperprior. The whole pipeline is shown in Figure 3.5.



Component	Symbol
Input Image	$\boldsymbol{x}$
Encoder	$f(\boldsymbol{x};\boldsymbol{\theta}_e)$
Latents	$\boldsymbol{y}$
Latents (quantized)	$\hat{\boldsymbol{y}}$
Decoder	$g(\hat{\boldsymbol{y}};\boldsymbol{\theta}_d)$
Hyper Encoder	$f_h(\boldsymbol{y};\boldsymbol{\theta}_{he})$
Hyper-latents	$\boldsymbol{z}$
Hyper-latents (quant.)	$\hat{\boldsymbol{z}}$
Hyper Decoder	$g_h(\hat{\boldsymbol{z}};\boldsymbol{\theta}_{hd})$
Context Model	$g_{cm}(\boldsymbol{y}_{<i};\boldsymbol{\theta}_{cm})$
Entropy Parameters	$g_{ep}(\cdot;\boldsymbol{\theta}_{ep})$
Reconstruction	$\hat{\boldsymbol{x}}$

Figure 3.5 *(Courtesy of the authors of [40], permissions to use this figure and its caption are given for this thesis): The combined model jointly optimizes an autoregressive component that predicts latents from their causal context (Context Model) along with a hyperprior and the underlying autoencoder. Real-valued latent representations are quantized (Q) to create latents ($\tilde{\boldsymbol{y}}$) and hyper-latents $\hat{\boldsymbol{z}}$), which are compressed into a bitstream using an arithmetic encoder (AE) and decompressed by an arithmetic decoder (AD). The highlighted region corresponds to the components that are executed by the receiver to recover an image from a compressed bitstream.*

The probability of $\hat{\boldsymbol{y}}$, given the learned parameters then becomes:

$$\begin{gathered} p_{\hat{y}}\left(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{z}}, \boldsymbol{\theta}_{hd}, \boldsymbol{\theta}_{cm}, \boldsymbol{\theta}_{ep}\right)=\prod_i\left(\mathcal{N}\left(\mu_i, \sigma_i^2\right) * \mathcal{U}\left(-\tfrac{1}{2}, \tfrac{1}{2}\right)\right)\left(\hat{y}_i\right) \\ \text{with } \mu_i, \sigma_i=g_{ep}\left(\boldsymbol{\psi}, \boldsymbol{\phi}_i ; \boldsymbol{\theta}_{ep}\right), \boldsymbol{\psi}=g_h\left(\hat{\boldsymbol{z}} ; \boldsymbol{\theta}_{hd}\right), \text{ and } \phi_i=g_{cm}\left(\hat{y}_{<i} ; \boldsymbol{\theta}_{cm}\right), \end{gathered} \tag{3.4}$$

where $\boldsymbol{\theta}_{hd}, \boldsymbol{\theta}_{cm}$ and $\boldsymbol{\theta}_{ep}$ are the learned parameters of the hyper-decoder, context model, and entropy parameters networks, respectively.

Without the zero-mean assumption of the Gaussian mixtures, the predicted distribution of $\hat{\boldsymbol{y}}$ would have smaller scales and would be more accurate, thus improves the compression rate. Additionally, combining the information stream from the context model with one from the hyper-autoencoder, in the authors' words, is beneficial from two perspectives:

- Context model can provide more information without giving up bitrate because it is autoregressive.
- The hyper-autoencoder can "look into the future" to provide more useful information.

Figure 3.6 visualizes the different results coming from different hyperprior approaches. By the trained model of each kind, the same input image is compressed, of which the latents of the highest entropy channel are visualized. The proposed model with a context module shows superior predictions of the latent values, consequently leading to the best entropy compression efficiency.

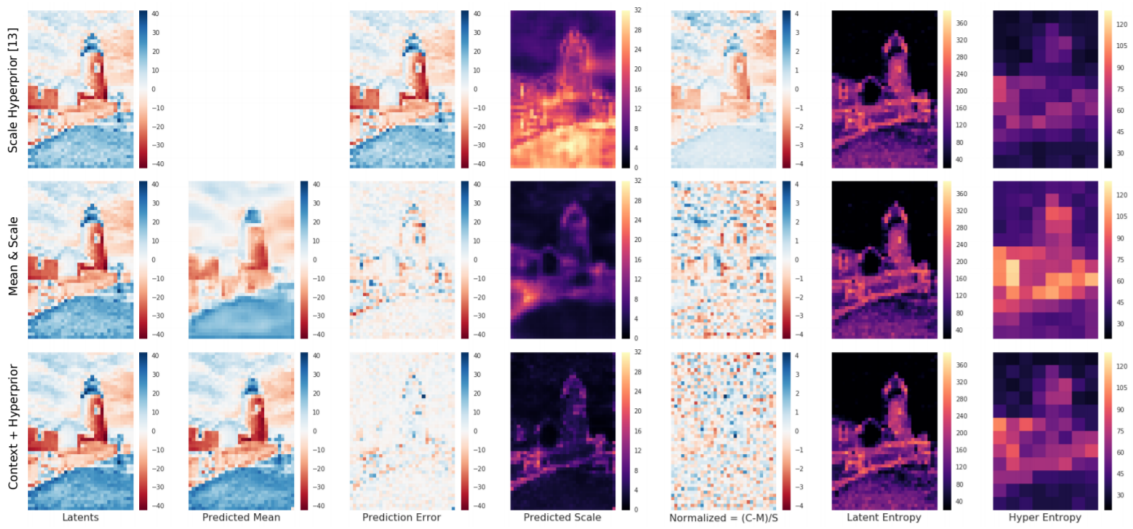


Figure 3.6 *(Courtesy of the authors of [40], permissions to use this figure and its caption are given for this thesis): Each row corresponds to a different model variant and shows information for the channel with the highest entropy. The visualizations show that more powerful models reduce the prediction error, require smaller scale parameters, and remove the structure from the normalized latents, which directly translates into a more accurate entropy model and thus higher compression rates. Their entropy model assumes that latents are conditionally independent given the hyperprior, which implies that the normalized latents, i.e. values with the predicted mean and scale removed, should be closer to independent and identically distributed Gaussian noise.*

3.2.3 The entropy model in the proposed ICM system

One major drawback of the proposed joint context and hyperprior model in subsection 3.2.2 is the causality inherited from the autoregressive context model, meaning the inference on this model has to be sequential. This makes the inference speed prohibitively slow since the model can not take advantage of parallel computing, resulting in an impractical solution. Because of that, the proposed ICM system in this thesis opts for a more practical variant of this model for its Entropy module, which is a simpler extension of the scale-hyperprior model described in subsection 3.2.1. This model transforms the hyperprior into mean and scale parameters, but unlike the joint context-hyperprior model, it does not employ the context model. The final pipeline is thus shown in Figure 3.1, where the *Encoder* and *Decoder* modules are described in section 3.1. Note that the proposed ICM pipeline can also be viewed as a modified version of the **"Mean & scale Hyperprior"** described in [40], with the main coder replaced by the auto-encoder in section 3.1.

3.3 Task networks

Since the proposed ICM system targets the image vision task-NNs as the main consumers, the high fidelity output images is not a requirement. Instead, task performance is prioritized. Good task performance can be achieved by imposing corresponding task loss $\mathcal{L}_{task}$ minimization on the proposed system training. In order to validate the proposed system, 2 different models are trained and evaluated for image compression targeting 2 different computer vision tasks: object detection with Faster R-CNN [51] and instance segmentation with Mask R-CNN [54]. The VCM Ad-hoc group also chooses these vision tasks and network architectures for study in the MPEG standardization activities.

The task network in the proposed pipeline (illustrated in Figure 3.1) is frozen, meaning no modification is allowed to the pre-trained model weights. The task loss term $\mathcal{L}_{task}$ is assigned to the training task loss of the respective task model. Gradients derived from the task loss differentiation are back-propagated to update the main coder. The task networks are frozen in this system is a practical aspect that enables the resulting system to be easily integrated to other workflows. Without this constraint, the ICM system might achieve even better compression efficiency due to the fact that the task network is further optimized, but the integration of the learned codec would require also the fine-tuning of the task network in the existing host workflows, therefore makes it less adoptive.

The task loss $\mathcal{L}_{task}$ is defined for each task object detection and instance segmentation by Equation 2.11 and Equation 2.12 of chapter 2, respectively.

3.4 Training strategy

Human-oriented image coding is often formularized as a rate-distortion optimization (RDO) problem:

$$J = R + \lambda \cdot D, \tag{3.5}$$

where J denotes the cost function, R is the expected rate loss and D is the expected distortion loss, and λ is the Lagrange multiplier governing the trade-off between them [1–3, 5, 6, 36–40]. The expected distortion is usually measured by the ℓ_2-norm distance between the input image $\boldsymbol{x}$ and output images $\hat{\boldsymbol{x}}$: $D = \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$. To reduce distortion D on the output, more information needs to be encoded, consuming more encoding bits, i.e. high bitrate R, and the other way around. The common way of performing RDO in NN-based methods is to find a certain set of values for λ (and other hyperparameters) for each working point so that the desired bitrate–distortion is achieved after a certain number of training iterations [36–40]. The search space for RDO is illustrated in Figure 3.7. The learned models are then saved as compression models for the bitrate that they achieved on the validation set.

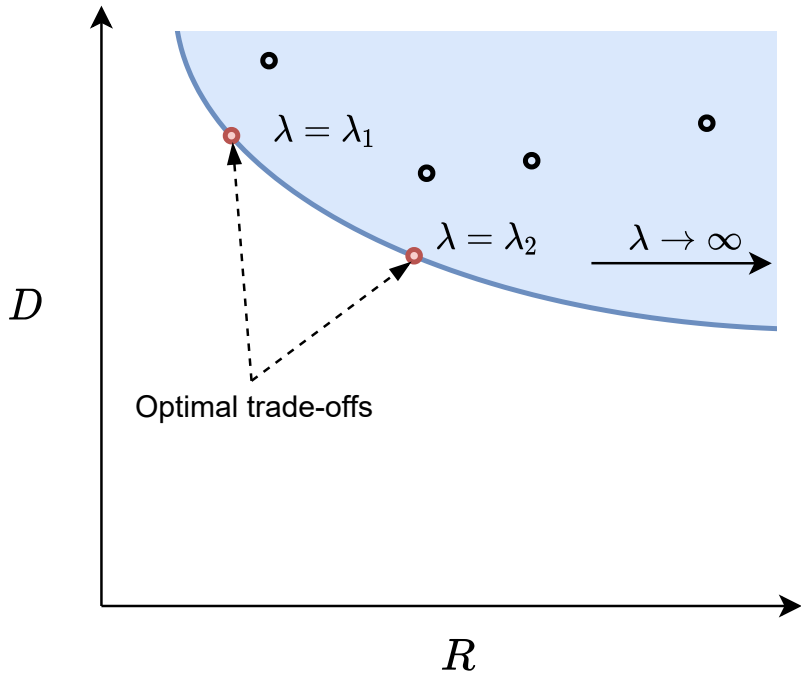


Figure 3.7 *The rate-distortion trade-off search space as described in [38]. The shaded region represents all the possible rate-distortion trade-offs over all possible parameter settings.*

Since the proposed ICM system is optimizing for task performance, it naturally extends the above mentioned conventional RDO approach by adding the task loss $\mathcal{L}_{task}$ to the "distortion" term D. The training is then formulated as a multi-task optimization, the general training loss function is given by:

$$\mathcal{L}_{total} = w_{rate}\mathcal{L}_{rate} + w_{mse}\mathcal{L}_{mse} + w_{task}\mathcal{L}_{task}, \tag{3.6}$$

where w_{rate}, w_{mse} and w_{task} are the scalar weights for each loss term $\mathcal{L}_{rate}, \mathcal{L}_{mse}$ and $\mathcal{L}_{task}$, respectively. The Mean Square Error (MSE) loss term is error between the input and output of the codec: $\mathcal{L}_{mse} = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$, where N denotes the mini-batch size. $\mathcal{L}_{task}$ and $\mathcal{L}_{rate}$ were defined and discussed earlier in section 3.3 and section 3.2, respectively.

3.4.1 Loss weighting strategy

Multi-task training loss such as Equation 3.6 can be optimized by simply predefining a set of scalar values for w_{rate}, w_{mse} and w_{task} that result in the desired minimization of each loss term. Another approach is to dynamically balance the losses based on different on-the-fly statistical analysis of the gradients or the losses themselves [48–50].

Rather than using fixed loss weights, the proposed ICM system uses a dynamic

loss weighting strategy for effective multitask training because of the following reasons:

- The competing nature of the loss terms makes their respective gradients to worsen the performance of the others'. It is critical to have the right balance between the objectives in each update, which is very challenging for fixed loss weighting due to its inflexibility.
- Exhaustive search for the optimal weights is very time-consuming [49].

However, the dynamic loss balancing techniques mentioned above are based on the statistics data (e.g. loss changing rate, prediction uncertainty), that are only visible in runtime. Therefore governing the trade-off between the objectives requires analysis and manipulation at low-level functions, which poses a challenge to those who wish to shift the balance between objectives. In image and video coding, the priorities of the objectives are different for each use case. For example, a traffic surveillance camera system may prefer a low latency transmission to the ultra-high video quality, thus it accepts the low bitrate and higher distortion trade-off. In contrast, a photo library software may prioritize better quality for its offline image labeling using a classification model, hence prefers low distortion with high bitrate trade-offs. The ability to control the trade-offs for multiple targeted bitrates, often referred to as "rate-control", is a feature well-desired by image and video codecs. To tackle all of the aforementioned requirements, the proposed loss weighting strategy is a dynamic loss weighting technique that offers better "rate-control" in a higher-level expression than the previous automatic loss balancing techniques, while still manages to handle the competing-losses balancing problem effectively. The general principles of the proposed strategy is discussed below, followed by two concrete examples.

General strategy: The task networks are trained on natural images, thus expect close-to-natural images as their input. In that light, the proposed ICM system trains a base model with only $\mathcal{L}_{mse}$ ($w_{task} = w_{rate} = 0, w_{mse} = 1$). The base model is capable of reconstructing images for decent task performance since the reconstructed images are very good resemblances of the input ones, imposed by low MSE. Then it fine-tunes the base model by gradually raising w_{rate} and w_{task}, which eventually 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, i.e. closer trade-off values to the convex hull in Figure 3.7. Consequently, the same training instance is able to achieve a new rate-task performance value for a different targeted bitrate after every iteration (see the results in section 5.2). In other words, in every iteration,

the model tries to obtain a trade-off value that is close to the optimal convex hull. Learning rate decay [58] is applied to keep the training stable, as the magnitude of $\mathcal{L}_{total}$ in Equation 3.6 tends to increase over time due to the raising loss weights. At inference time, the 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.

Example 1: In this example, the evolutions of the loss weights throughout training iterations happen in five phases. They are illustrated in Figure 3.8. Each phase of the training is intuitively designed in accordance with the user's intentions:

- Phase 1: To train the base model with only $\mathcal{L}_{mse}$ for decent task performance. The weights for the other losses are 0s.
- Phase 2: $\mathcal{L}_{task}$ starts to contribute to the gradients due to the increasing w_{task}.
- Phase 3: $\mathcal{L}_{rate}$ is gradually introduced to the training.
- Phase 4: To focus on enhancing the task performance by increasing w_{task} while keeping w_{rate} unchanged.
- Phase 5: The system is now stable. Search for the best trade-offs of the two main objectives ($\mathcal{L}_{task}$ and $\mathcal{L}_{rate}$) in different policies imposed by the increasing gradient flow dominance of them. For example, since the w_{rate} is high in this phase, the network is forced to heavily compress the input data thus achieve a low bitrate, while high w_{task} at the same time force the system to store the information that is critical for the task-NN under the given bitrate constraint.

Example 2: Another example of the loss weighting techniques that uses the proposed principles is illustrated by Figure 3.9. The strategy is designed as followings:

- Similar to the previous loss weighting example, let $\mathcal{L}_{mse}$ dominate the gradient flow in the first phase, the ease down its influence.
- $\mathcal{L}_{task}$ and then $\mathcal{L}_{rate}$ gradually get improved after the "warm-up" phase.
- $\mathcal{L}_{task}$ stops increasing its impact after a certain number of iterations, therefore, leaves room for the bitrate $\mathcal{L}_{rate}$ improvement.
- $\mathcal{L}_{rate}$ keeps increasing its influence till the end of the training, effectively pushing for the best bitrate–task performance trade-offs.

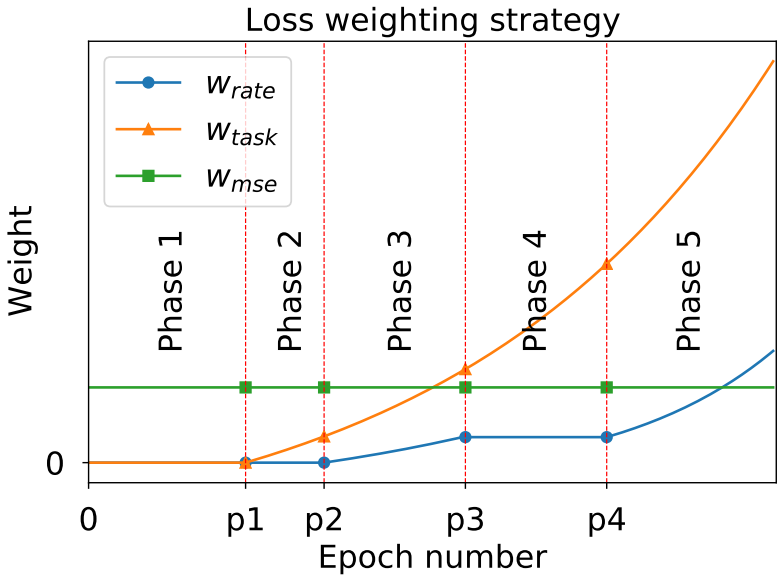


Figure 3.8 *Example 1: Loss weights evolution over iterations. There are 5 phases in this strategy, separated by the vertical lines.*

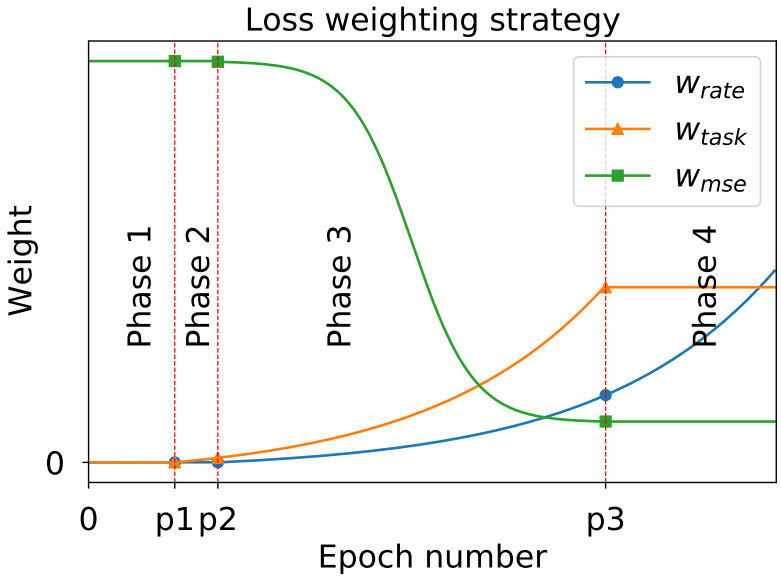


Figure 3.9 *Example 2: Loss weights evolution over iterations. There are 4 phases in this strategy, separated by vertical lines.*

3.4.2 Quantization approximation

The quantization step $\boldsymbol{Q}(\cdot)$ makes the gradients with respect to the parameters of the encoder become zero almost everywhere. In order to enable end-to-end training for the proposed ICM system, the quantization approximation [59] is applied, which substitutes the quantization step with additive uniform noise. During the training

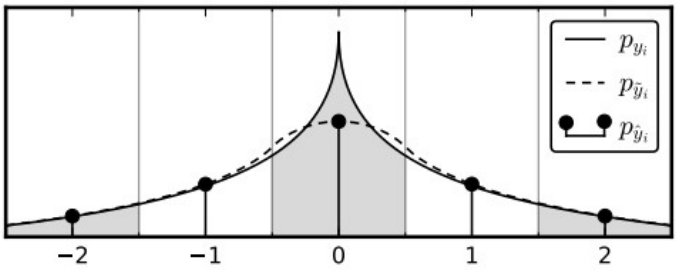


Figure 3.10 *(Courtesy of the authors of [59], permissions to use this figure and its caption are given for this thesis): $p_{\hat{y}_i}$ is densities of the quantized y_i ($\hat{y}_i$). $p_{\tilde{y}_i}$ is the continuous approximation of the mass in each quantization bin.*

process, the quantization step is relaxed by this technique, and it is switched back on during compression (inference stage). The approximated quantization of an element y_i is denoted by a tilde: $\tilde{y}_i = y_i + \Delta y_i$, with $\Delta y_i \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2})$. Then the density function is

$$p_{\tilde{y}_i} = p_{y_i} * \mathcal{U}(-\frac{1}{2}, \frac{1}{2}), \tag{3.7}$$

where "$*$" is the continuous convolution operation. The example densities are illustrated in Figure 3.10.

In section 5.4 of chapter 5, the effect of this technique on the proposed codecs is evaluated.

4 Experiments

The framework in chapter 3 is evaluated on two tasks: instance segmentation and object detection as these are ones of the key vision tasks that receive the attention of the VCM Ad-hoc group [24]. Both of the neural network models for the two tasks use Resnet-50-FPN backbones [28]. The task networks are pre-trained on COCO dataset [60]. The pre-trained models are provided by Torchvision[1][61], and the rest of the framework is implemented with Pytorch 1.5 [62].

The uncompressed dataset Cityscapes (fine annotations) [63] is used for training the proposed models (*train* subset), and for codec evaluation (*val* subset). The *val* (validation) subset consisting of 500 images is used for the evaluations of the VVC codec and the proposed learned codecs targeting 2 different tasks: object detection and instance segmentation. Since the task models are pre-trained on the COCO dataset, only the results for the common classes of the 2 datasets are evaluated, which are: *car, person, bicycle, bus, truck, train, motorcycle*. The environmental configurations are shown in Table 4.1.

Table 4.1 *Environmental configurations*

Hardware configurations	Software
CPU: Intel Core i9-9940X GPU: NVIDIA RTX 2080Ti 11GB (×2)	OS: Ubuntu 18.04.4 LTS Pytorch: 1.5.0 Torchvision: 0.6.0 CUDA: 10.2 Python: 3.8.3 64bit

4.1 Pareto front

In multi-objective optimization, the Pareto front (or Pareto frontier) [64] is used for representing the most efficient allocations that are not strictly dominated by any other candidates. In other words, it is the set of best solutions for every given criterion. An example of a Pareto front is given by Figure 4.1. For a fair and practical comparison, the candidates from the proposed method and the baseline should be the best possible ones from both sides, hence the Pareto front is used as the algorithm to select the candidates for the comparison of the results.

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

[2]Available at `https://commons.wikimedia.org/wiki/File:Front_pareto.svg`

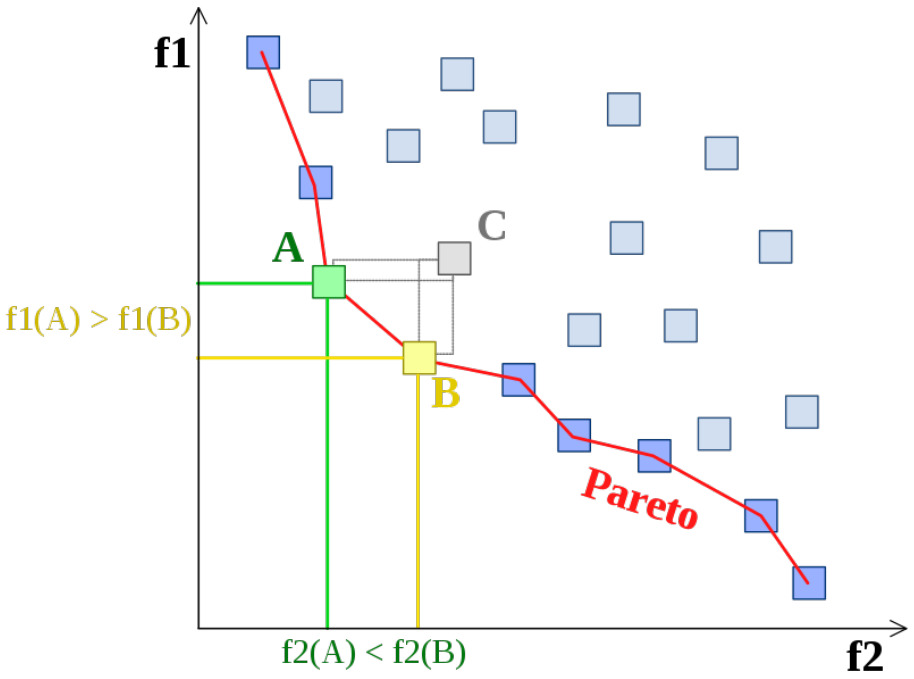


Figure 4.1 *(Courtesy of Johann Dréo[2] under the CC BY-SA 3.0 license) An example of Pareto front. The boxed points represent feasible choices (candidates), and smaller values are preferred to larger ones. Point C is not on the Pareto frontier because it is dominated by both point A and point B. Points A and B are not strictly dominated by any other, and hence lie on the frontier.*

4.2 Evaluation baseline - VVC codec anchors

The baseline performance is established by using the reference software VTM-8.2 [65], All-Intra configuration of the state-of-the-art codec standard VVC, under JVET common test conditions (CTC) [66]. The Quantization Parameter (QP) values are commonly used for rate-control, i.e. achieving different bitrate-distortion trade-off, in traditional codecs such as VVC. Considering the task networks are state-of-the-art models which are trained to be robust to the input image scales, the scaling factors of the input images are also taken into account as rate-control options of VVC in an ICM system. Thus, in addition to using the QP parameters to achieved different bitrates, the VVC codec in this evaluation also takes advantage of the scale-robustness of the task network to create better baseline anchors. For example, the object detection task network may produce the same level of task performance when using inputs that are downsampled by 2 in both width and height, resulting in a 4 times bitrate reduction while preserving the same task performance. Therefore, there are two set of settings for rate-control with VVC in this ICM benchmark: QP $\in \{22, 27, 32, 37, 42, 47, 52\}$ and resolution $\in \{100\%, 75\%, 50\%, 25\%\}$, making up a total of 28 combinations. The resolution represents the downsampled size of the images in both width and height, e.g. a resolution 50% image has its width

and height equal to half of them of the 100% resolution image, which means it has a 4 times less number of pixels. The validation set (*val* subset of Cityscapes) is compressed using these 28 configurations, resulting in 28 versions. Each version is evaluated for bitrate and task performance as an anchor point. A summary of the anchor generation for each combination of QP and resolution is shown in Figure 4.2.

The Pareto front (discussed in section 4.1) of the anchors is chosen to be the baseline for ICM targeting the corresponding task network, visualized in Figure 5.1.

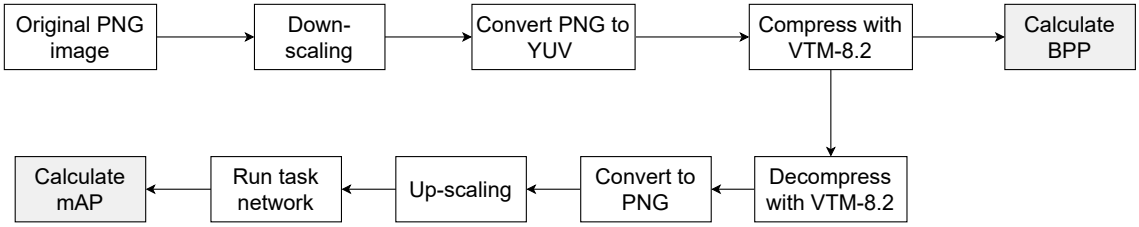


Figure 4.2 *Baseline anchor generation for each set of QP and resolution configurations of the VVC codec. Note that PNG [67] is a lossless compression, therefore the PNG decoded data is still considered as uncompressed images.*

4.3 System setup for training and evaluation of the learned codecs

4.3.1 Training setup

The proposed systems for two NN-based codecs targeting the two task networks are trained on the *train* subset of Cityscapes (fine annotations) dataset, which consists of 2975 2048 $\times$ 1024 images and pixel-level annotations for both object detection and instance segmentation. Each of the codecs has about 1.5 millions trainable parameters. Because of the GPU memory limitation, the auto-encoder (codec) and the task network are placed into two separate GPUs, in each system setup. The codecs were optimized using gradient-based Adam optimizer [68] with batch size of 1 and learning rate starting at 10^{-5}. The learning rate is scheduled to decrease by 0.005% every epoch. The two codecs were trained for 984 and 993 epochs for object detection and instance segmentation, respectively, in roughly 2 weeks.

Loss weighting strategy In these experiments, the loss weight values w_{mse}, w_{task} and w_{rate}, discussed earlier in subsection 3.4.1, are modeled as functions of the training epoch number that follow the example strategy number 1 illustrated by

Figure 3.8. Concretely, the weight values at epoch number e is given by:

$$
\begin{aligned}
w_{mse} &= 1, \\
w_{task} &= \begin{cases} 0, & e < p_1 \\ 4f_w(e - p_1, 1.01), & e >= p_1 \end{cases}, \\
w_{rate} &= \begin{cases} 0, & e < p_2 \\ 2f_w(e - p_2, 1.01), & p_2 <= e < p_3 \\ 2f_w(e - p_3, 1.01), & p_3 <= e < p_4 \\ 2f_w(e - p_4, 1.02), & e >= p_4 \end{cases},
\end{aligned}
\tag{4.1}
$$

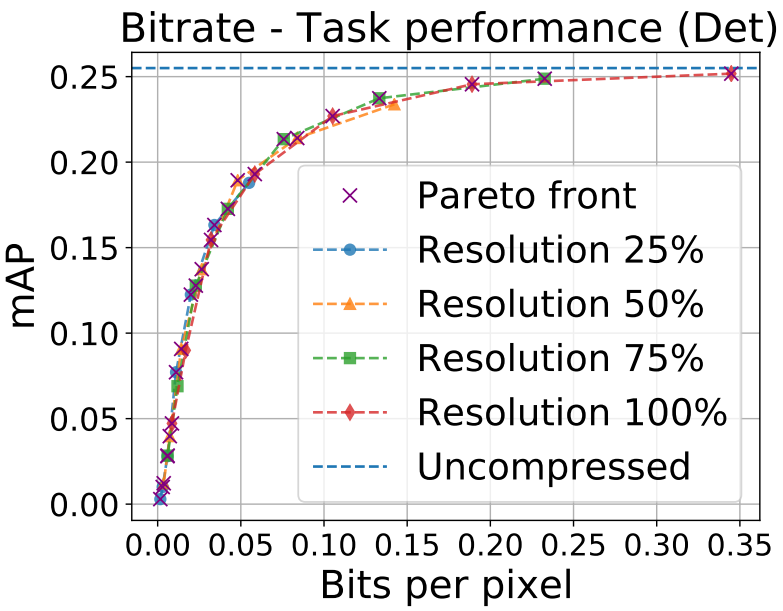
where $(p_1, p_2, p_3, p_4) = (50, 75, 120, 165)$ and $f_w(x, a) = 10^{-3}(a^{x-1})$

4.3.2 Evaluation setup

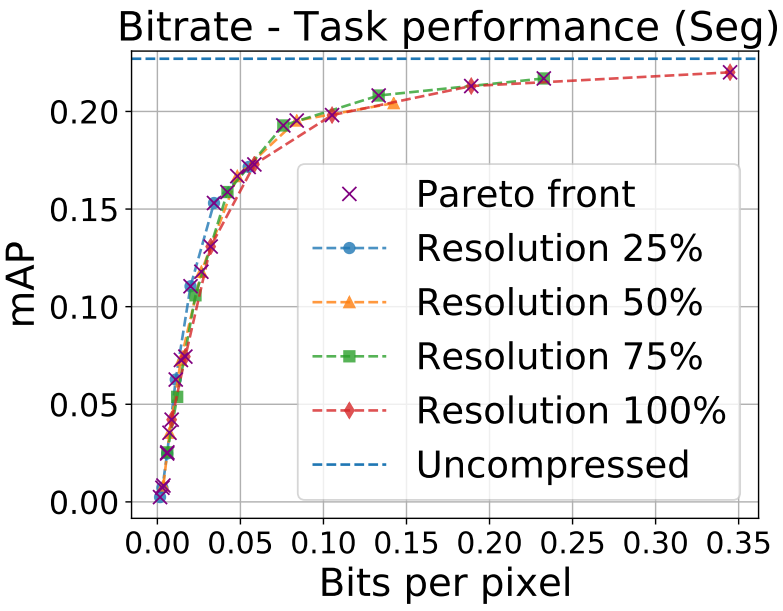
After every epoch of training, the trained model is evaluated on the validation set and its states are saved as a checkpoint. The model is evaluated on 500 images of the *val* (validation) subset of Cityscapes (fine annotations). The evaluation result for every epoch comprises the average bitrate and the average task performance over the whole validation set. The metric for bitrate is Bits Per Pixel (BPP), calculated by dividing the length of the compressed bitstream by the number of pixels in the original image, which in this case is $2048 \times 1024 = 2097152$ pixels for every image in the validation set. The task performance metrics, for both object detection and instance segmentation, are mean Average Precision (mAP) of different IoU thresholds in range [0.5:0.05:0.95] as described in [63]. The results for every epoch is saved as candidates for Pareto set selection (discussed in section 4.1) in later comparison with the baseline. Note that the reported bitrates for comparison with the baseline in chapter 5 are the actual bitrates arising from the length of the compressed bitstream using the learned codecs.

5 Evaluation results

5.1 Baseline performance - VVC anchors



(a) Object detection



(b) Instance segmentation

Figure 5.1 *Baseline performance given by the VVC codec for two tasks: object detection and instance segmentation.*

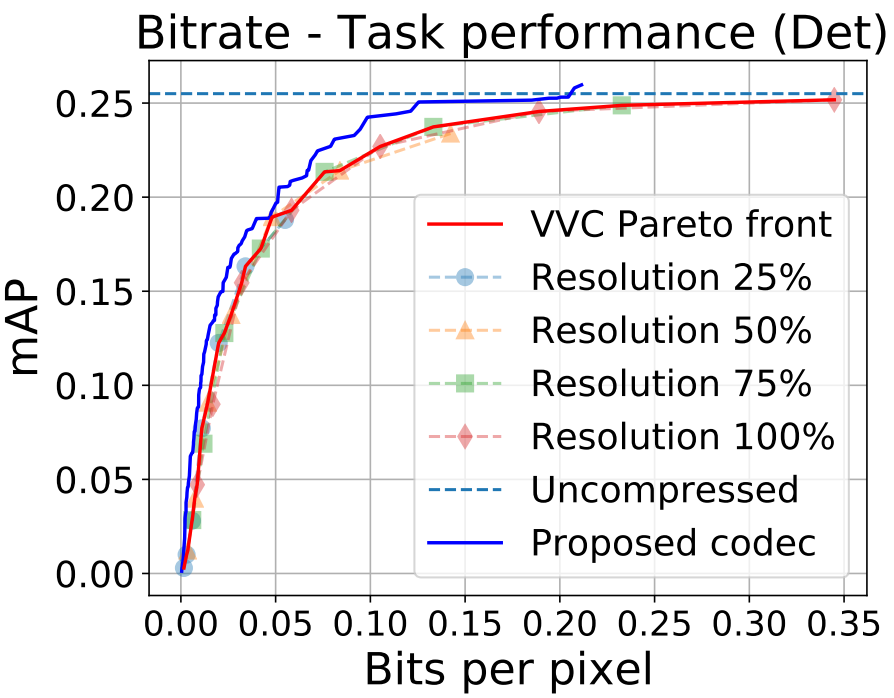
Figure 5.1 shows the anchors and their Pareto fronts, which are chosen to be

the baselines for the targeted tasks. The baseline evaluation process is described in the previous chapter. As can be seen from the figures, most of the data points are included in the Pareto sets. Additionally, it is clearly shown in the evaluation of task instance segmentation, the curve of resolution 75% lies above the one of resolution 100% (i.e. it has better trade-offs) in bitrate range $[0.05, 0.25]$, while the curve of resolution 25% is the most efficient in bitrate range $[0, 0.05]$. In the evaluation of object detection, the right-most green anchor points, corresponding to QP option 22 of resolution 75%, has almost the same task performance as the one from resolution 100%, QP 22 (the right-most red anchor point), while saving about 40% of bitrate. The above analysis proves that the *resolution* factor, in addition to *QP*, is also a valuable bitrate control option for ICM. Therefore the Pareto fronts of these anchor points are closer representations of the optimal solutions for ICM using a traditional codec, namely VVC.

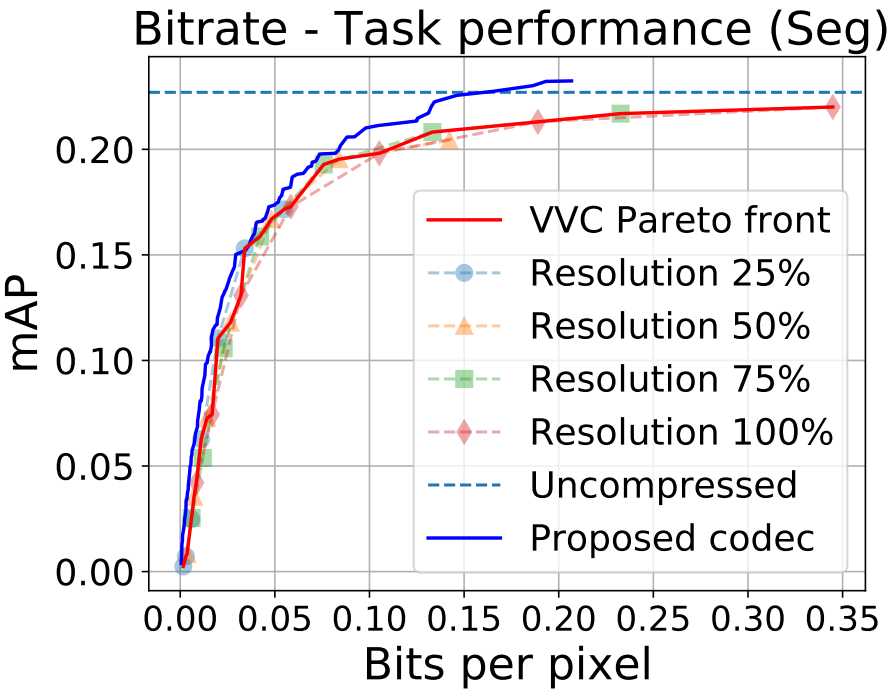
Lastly, it should be noticed that due to the compression distortions of the traditional codec, the task performance on the compressed data is always worse than the task performance of the uncompressed data, which is represented by the horizontal lines on the top of the two figures.

5.2 Compression efficiency in comparison against the baseline

Figure 5.2 shows that the proposed codecs outperform their baselines at almost every given bitrate, i.e. the task performance on the dataset compressed by the proposed codecs is almost always better than the one coming from VVC compression, indicated by the blue curves of proposed codecs being above the red curves of VVC standard codec, in both of the figures for the two tasks. Interestingly, while the traditional codec VVC introduces distortions that are harmful to the task performance, even with configurations for high-quality output such as QP 22 of resolution 100%, the learned codec can improve the task performance if the bitrate budget allows, as shown on the figures, where the blue curves manage to rise above the horizontal lines of uncompressed task performance, for example, when the bitrates are higher than 0.15 bpp, the instance segmentation performance of the data compressed by the learned codec is better than that of the uncompressed data. It is worth mentioning that at the peak performance, the proposed codecs still consume less than 70% amount of bitrate for the QP 22 of resolution 100%, in both tasks.



(a) Object detection



(b) Instance segmentation

Figure 5.2 *Performance comparison of the proposed codecs against the baselines for the corresponding tasks*

In traditional image compression, one way to compare the efficiency of two codecs is to calculate the Bjøntegaard Delta Rate (BD-Rate)[69] with respect to Peak Signal

To Noise Ratio (PSNR). This metric reports the average difference between the two Rate-Distortion curves. Within the ICM landscape, the PSNR metric is replaced by the task performance, as well as the Rate–Distortion curves are replaced by the Rate–Task performance curves such as the ones in Figure 5.2. For this thesis, the main comparisons would be between the Pareto fronts of the same targeted task. As a common practice, each anchor point on the baseline curves is matched with a point of the comparing curves (i.e. the curves from the proposed codecs) that has the closest bitrate. Then the BD-Rate values are calculated on the overlapping range in terms of bitrate between the pairs of curves made up from the matched points.

The BD-Rate values of the curves in Figure 5.2 are shown in Table 5.1. These values represent the average bitrate savings of the proposed codecs in comparison with the VVC baselines. For instance, the proposed codec for task object detection on average consumes 37.87% less bitrate for a similar level of task performance compared to the Pareto front of VVC anchors. Similarly, the proposed codec for task instance segmentation also outperforms the VVC codec by 32.90% of bitrate saving. As a reference, although not directly comparable, for instance segmentation using the same validation data (*val* subset of Cityscapes) and same task-NN architecture (Mask R-CNN with Resnet-50-FPN backbone), the authors of [30] report up to 9.95% of bitrate saving compared to VVC codec for the following QPs: 12, 17, 22 and 27 of resolution 100%. Note that they use different QP values and their model is pre-trained directly for Cityscapes on its *train* subset [70].

Table 5.1 *Bjøntegaard Delta Rate (BD-Rate) with respect to task performance of the proposed codecs against the baseline and additional VVC anchors of different resolution settings.*

		Resolution			
Targeted task	Pareto front (baseline)	100%	75%	50%	25%
Detection	-37.87%	-32.86%	-33.89%	-34.76%	-38.76%
Segmentation	-32.90%	-33.77%	-28.58%	-29.98%	-28.04%

The average encoding time of a 2048×1024 image in the Cityscapes validation set is approximately 0.15 seconds with a batch size of 1. As a reference, the VVC encoding runtime for the same resolution (100%) validation image is about 126 seconds on a cluster with Intel Xeon Gold 6154 CPUs (Table 5.2).

Table 5.2 *VVC codec encoding time using VTM-8.2 reference software [65]*

	Resolution			
Encoding time	100%	75%	50%	25%
Overall (Sec.)	376777.6	251454.6	142702.3	48529.95
Average (Sec.)	125.5925	83.8182	47.56743	16.17665

5.3 Visual output inspection

In this section, a few examples of the output images from the proposed codec for instance segmentation are examined to gain important insights into the efficiency of the method. Figure 5.3 shows the decompressed images by the learned codec, which are given to the task network (instance segmentation) for task performance evaluation. The images on the same row are the outputs of the codec for different targeted bitrate, given the same input image. These outputs show that the system has learned to compress more aggressively on the regions that are not too important to the task network. In particular, the shapes and edges of the pedestrians or vehicles on the street are better-preserved than other regions on the images, since those are the instances that the task network has to identify. The learned codec, by effectively suppressing the details for non-important regions, manages to consume less bitrate for the compressed data, while preserving the task performance, eventually leads to a better compression efficiency than that of the traditional VVC standard codec.

5.4 Ablation study on the quantization approximation accuracy

As mentioned in subsection 3.4.2, a relaxed quantization is applied during the training process of the model. In practice, to avoid extra complexity and reduce training time, the experiments in this thesis after every epoch of training also report an estimated bitrate with approximated quantizations instead of running an actual compression that outputs the binary bitstream. The preliminary Pareto fronts are selected by evaluating the estimated bitrates of their corresponding checkpoints. The checkpoints that are selected to be on the preliminary Pareto front then are re-evaluated for the accurate bitrate for the benchmark reports. The differences between the estimated bitrates with relaxed quantization and the actual bitrates obtained by running compression with quantization are insignificant, as shown in Figure 5.4. In the lower bitrate range, the estimated bitrates and the actual bitrates are almost identical, for both codecs targeting the two tasks.

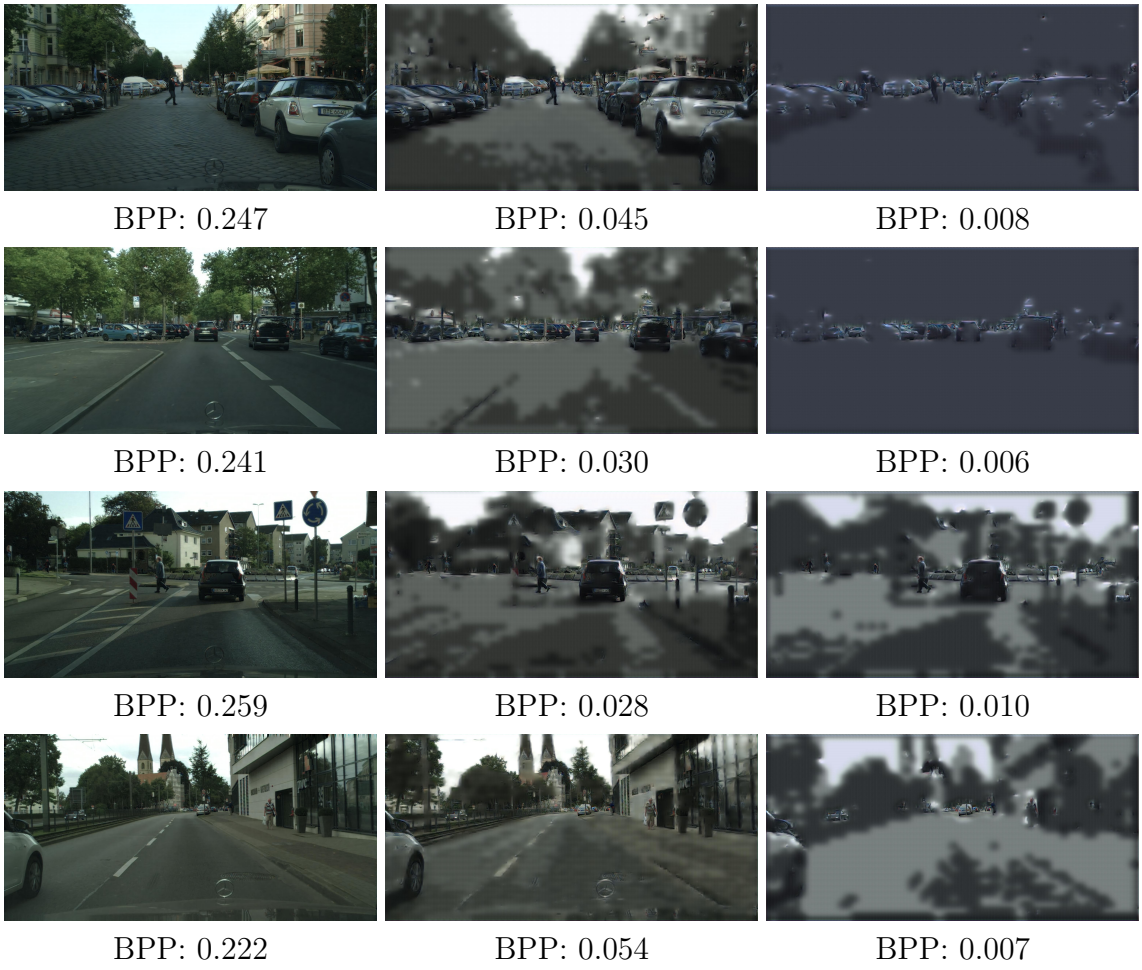
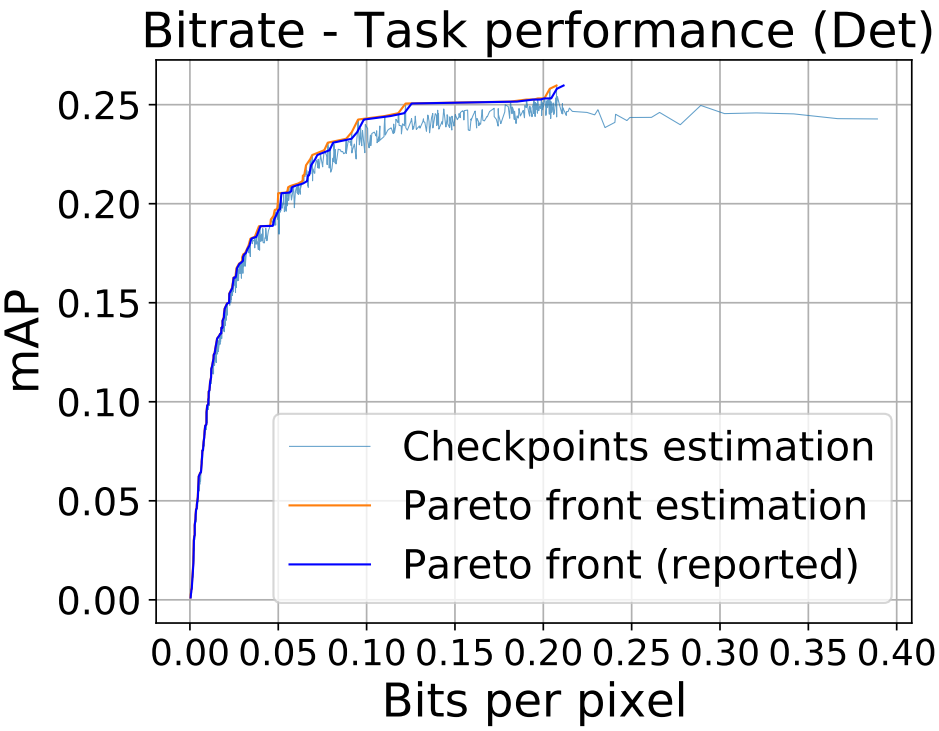
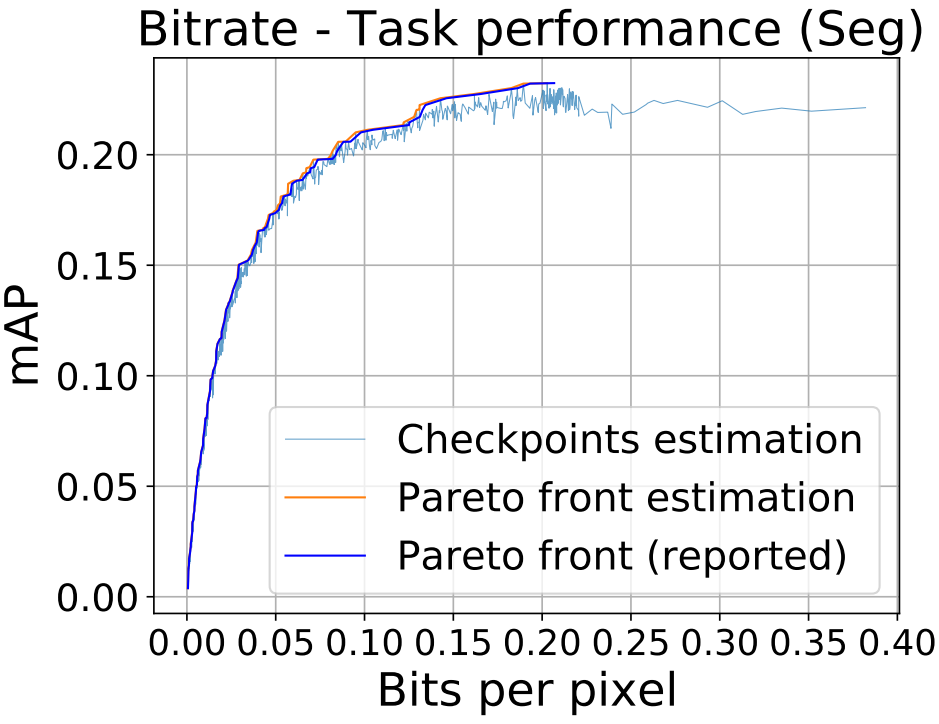


Figure 5.3 *Decoded outputs for instance segmentation in different bitrate ranges. The lower bitrate, the more suppressed background to reserve the bitrate budget for the more important regions that include the objects the task network is trying to detect. In those images that contain many important object instances, the backgrounds are almost flattened out.*



(a) Object detection



(b) Instance segmentation

Figure 5.4 *Estimated bitrates using quantization approximation versus the actual bitrates. "Pareto front (reported)" denotes the curves that are obtained by running real compressions and were reported for comparison with the VVC anchors in chapter 5.*

6 Discussion and summary

In this thesis, the downsides of using traditional video coding standards for machines to machines communication are addressed. They mostly come from the fact that traditional codecs are designed for human-consumption. That makes the traditional codecs inefficient when the end-users are vision tasks, both in terms of bitrate and task-performance. Next, an overview of prior work aiming to tackle those problems is given. At the time when this thesis is in progress, the prior arts are mainly focused on modifying or enhancing the traditional paradigms for better task performance.

In chapter 3, a novel ICM (Image coding for machines) system is proposed. In this system, the codec is trained to directly improve the performance of the task networks. To the best of the author's knowledge, this is the first time an end-to-end neural network-based system is proposed to address the Rate–Task performance optimization considering machines as the only consumers. Additionally, an effective training strategy for multi-task optimization is also introduced. The efficiency of the new proposed system is compared to that of the state-of-the-art traditional codec, VVC/H.266 (chapter 5). In this comparison, the proposed method demonstrates its superior efficiency when it comes to machines-targeted image coding by outperforming the traditional codec VVC/H.266. Moreover, the effect of the quantization approximation is also studied, it shows that the additive uniform noise can closely model the quantization effect on the latent representation. The experimental results reveal a few key points of the proposed method:

- **Coding efficiency:** The VVC baseline performance evaluation in this thesis are designed to take advantage of the spatial scaling robustness of the targeted task networks, on top of the conventional bitrate-control technique using Quantization parameters (QP). Even when compared to that baseline, the Rate–Task performance of the proposed method on average still saves 37.87% of bitrate for the same object detection performance with Faster R-CNN and 32.90% of bitrate for the same instance segmentation performance with Mask R-CNN.

- **Codec that acts as an enhancing module:** Instead of causing distortions that deteriorate the task performance as in traditional codecs, the results of the proposed codecs show that they are able to enhance the task performance on uncompressed data while significantly reduce the bitrate consumption. This is also proof of the inefficiency of traditional codecs in ICM use cases.

- **Coding complexity:** With a modest model size of 1.5M parameters, the

proposed codecs can compress image data at a practical speed (roughly 0.15 seconds per 2K image).

- **Compatibility:** Since the proposed method aims to completely replace the traditional codecs in ICM, the trained codecs can be seamlessly integrated into other pipelines without modifying the other components, e.g. fine-tuning the task networks in the existing workflow.
- **Further explorations:** The reconstructed output of the proposed codec indicate that although the codecs are trained for specific task networks, it still learns to preserve the critical information that is beneficial to different tasks or different architectures. On the other hand, there are likely correlations between the features needed by different tasks, which implies that compressing a generic bitstream for multiple tasks is more efficient than compressing the bitstreams separately. These are subjects for future work.

The proposed ICM system and the multi-task training techniques proposed in this thesis are highly adaptive and configurable. It can be easily utilized in other work targeting different objectives, or in the scope of ICM, targeting different task networks. As for further developments, this work opens new interesting research directions, such as the multitask bitstream compression and task features correlations study mentioned earlier. The auto-encoder architecture is also a promising topic since a rather small and simple one as proposed in this thesis already works out very well.

References

[1] T. Wiegand et al. “Overview of the H.264/AVC video coding standard”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (July 2003). Conference Name: IEEE Transactions on Circuits and Systems for Video Technology, pp. 560–576. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2003.815165.

[2] Gary J. Sullivan et al. “Overview of the High Efficiency Video Coding (HEVC) Standard”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012). Conference Name: IEEE Transactions on Circuits and Systems for Video Technology, pp. 1649–1668. ISSN: 1558-2205. DOI: 10.1109/TCSVT.2012.2221191.

[3] Benjamin Bross et al. “Versatile Video Coding (Draft 8)”. In: *Joint Video Experts Team (JVET), Document JVET-Q2001* (Jan. 2020).

[4] James Bankoski, Paul Wilkins, and Yaowu Xu. *VP8 Data Format and Decoding Guide*. URL: https://tools.ietf.org/html/rfc6386 (visited on 10/27/2020).

[5] G. Hudson et al. “JPEG at 25: Still Going Strong”. In: *IEEE MultiMedia* 24.2 (2017), pp. 96–103.

[6] D. S. Taubman and M. W. Marcellin. “JPEG2000: standard for interactive imaging”. In: *Proceedings of the IEEE* 90.8 (2002), pp. 1336–1357.

[7] A. Bilgin et al. “An Overview of JPEG-2000”. In: *Data Compression Conference*. Los Alamitos, CA, USA: IEEE Computer Society, Mar. 2000, p. 523. DOI: 10.1109/DCC.2000.838192. URL: https://doi.ieeecomputersociety.org/10.1109/DCC.2000.838192.

[8] *WebP: Compression Techniques*. Google. URL: https://developers.google.com/speed/webp/docs/compression (visited on 05/18/2020).

[9] Fabrice Bellard. *BPG Image format*. URL: https://bellard.org/bpg/ (visited on 10/05/2020).

[10] *Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper*. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (visited on 10/25/2020).

[11] Mariusz Bojarski et al. “End to End Learning for Self-Driving Cars”. In: *arXiv:1604.07316 [cs]* (Apr. 25, 2016). version: 1. arXiv: 1604.07316. URL: http://arxiv.org/abs/1604.07316 (visited on 10/18/2020).

[12] Vinod Nair and Geoffrey Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair". In: Proceedings of ICML. Vol. 27. June 16, 2010, pp. 807–814.

[13] Jürgen Schmidhuber. "Deep Learning in Neural Networks: An Overview". In: *Neural Networks* 61 (Jan. 1, 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. URL: http://www.sciencedirect.com/science/article/pii/S0893608014002135 (visited on 10/29/2020).

[14] Y. Lecun et al. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. ISSN: 1558-2256. DOI: 10.1109/5.726791.

[15] Vincent Dumoulin and Francesco Visin. *A Guide to Convolution Arithmetic for Deep Learning*. Jan. 11, 2018. arXiv: 1603.07285 [cs, stat]. URL: http://arxiv.org/abs/1603.07285 (visited on 10/29/2020).

[16] V.K. Goyal. "Theoretical foundations of transform coding". In: *IEEE Signal Processing Magazine* 18.5 (Sept. 2001). Conference Name: IEEE Signal Processing Magazine, pp. 9–21. ISSN: 1558-0792. DOI: 10.1109/79.952802.

[17] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (July 1948). Conference Name: The Bell System Technical Journal, pp. 379–423. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[18] Wikimedia Commons. *File:Arithmetic encoding.svg — Wikimedia Commons, the free media repository*. [Online; accessed 1-November-2020]. 2020. URL: %5Curl%7Bhttps://commons.wikimedia.org/w/index.php?title=File:Arithmetic_encoding.svg&oldid=490105139%7D.

[19] Ling-Yu Duan et al. "Overview of the MPEG-CDVS Standard". In: *IEEE Transactions on Image Processing* 25.1 (Jan. 2016), pp. 179–194. ISSN: 1941-0042. DOI: 10.1109/TIP.2015.2500034.

[20] L. Duan et al. "Compact Descriptors for Video Analysis: The Emerging MPEG Standard". In: *IEEE MultiMedia* 26.2 (Apr. 2019), pp. 44–54. ISSN: 1941-0166. DOI: 10.1109/MMUL.2018.2873844.

[21] Ling-Yu Duan et al. "Video Coding for Machines: A Paradigm of Collaborative Compression and Intelligent Analytics". In: *IEEE Transactions on Image Processing* 29 (2020), pp. 8680–8695. ISSN: 1941-0042. DOI: 10.1109/TIP.2020.3016485.

[22] J. Lin et al. “HNIP: Compact Deep Invariant Representations for Video Matching, Localization, and Retrieval”. In: *IEEE Transactions on Multimedia* 19.9 (Sept. 2017), pp. 1968–1983. ISSN: 1941-0077. DOI: 10.1109/TMM.2017.2713410.

[23] B. Sun et al. “CDVA/VCM: Language for Intelligent and Autonomous Vehicles”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020 IEEE International Conference on Image Processing (ICIP). Oct. 2020, pp. 3104–3108. DOI: 10.1109/ICIP40778.2020.9190735.

[24] Yuan Zhang, Manouchehr Rafie, and Shan Liu. *Use Cases and Requirements for Video Coding for Machines*. Oct. 16, 2020. URL: http://wg11.sc29.org/ (visited on 11/22/2020).

[25] Z. Chen et al. “Data Representation in Hybrid Coding Framework for Feature Maps Compression”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020 IEEE International Conference on Image Processing (ICIP). Oct. 2020, pp. 3094–3098. DOI: 10.1109/ICIP40778.2020.9190843.

[26] N. Yan et al. “Semantically Scalable Image Coding With Compression of Feature Maps”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020 IEEE International Conference on Image Processing (ICIP). Oct. 2020, pp. 3114–3118. DOI: 10.1109/ICIP40778.2020.9191184.

[27] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). ISSN: 1063-6919. June 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[28] T. Lin et al. “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). July 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[29] K. Fischer, C. Herglotz, and A. Kaup. “On Intra Video Coding And In-Loop Filtering For Neural Object Detection Networks”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020 IEEE International Conference on Image Processing (ICIP). Oct. 2020, pp. 1147–1151. DOI: 10.1109/ICIP40778.2020.9191023.

[30] Kristian Fischer et al. “Video Coding for Machines with Feature-Based Rate-Distortion Optimization”. In: *IEEE 22nd International Workshop on Multimedia Signal Processing* (Sept. 2020), p. 6.

[31] Benoit Brummer and Christophe 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)*. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Seattle, WA, USA: IEEE, June 2020, pp. 629–633. ISBN: 978-1-72819-360-1. DOI: 10.1109/CVPRW50498.2020.00090. URL: https://ieeexplore.ieee.org/document/9150597/ (visited on 09/21/2020).

[32] Charles Buysschaert et al. *JPEG XS, a New Standard for Visually Lossless Low-Latency Lightweight Image Coding System*. URL: http://ds.jpeg.org/whitepapers/jpeg-xs-whitepaper.pdf (visited on 11/22/2020).

[33] Zhao Wang, Ru-Ling Liao, and Yan Ye. “Joint Learned and Traditional Video Compression for P Frame”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Seattle, WA, USA: IEEE, June 2020, pp. 560–564. ISBN: 978-1-72819-360-1. DOI: 10.1109/CVPRW50498.2020.00075. URL: https://ieeexplore.ieee.org/document/9151080/ (visited on 10/07/2020).

[34] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016-06, pp. 1747–1756. URL: http://proceedings.mlr.press/v48/oord16.html.

[35] Tim Salimans et al. “PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications”. In: *arXiv:1701.05517 [cs, stat]* (Jan. 19, 2017). arXiv: 1701.05517. URL: http://arxiv.org/abs/1701.05517 (visited on 09/25/2020).

[36] F. Mentzer et al. “Practical Full Resolution Learned Lossless Image Compression”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). June 2019, pp. 10621–10630. DOI: 10.1109/CVPR.2019.01088.

[37] Sheng Cao, Chao-Yuan Wu, and Philipp Krähenbühl. “Lossless Image Compression through Super-Resolution”. In: *arXiv:2004.02872 [cs, eess]* (Apr. 6, 2020). arXiv: 2004.02872. URL: http://arxiv.org/abs/2004.02872 (visited on 09/21/2020).

[38] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. "End-to-end optimized image compression". In: *Int'l Conf on Learning Representations (ICLR)*. Toulon, France, Apr. 2017. URL: https://arxiv.org/abs/1611.01704.

[39] Johannes Ballé et al. "Variational image compression with a scale hyperprior". In: *arXiv:1802.01436 [cs, eess, math]* (May 1, 2018). arXiv: 1802.01436. URL: http://arxiv.org/abs/1802.01436 (visited on 09/21/2020).

[40] David Minnen, Johannes Ballé, and George D Toderici. "Joint Autoregressive and Hierarchical Priors for Learned Image Compression". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 10771–10780. URL: http://papers.nips.cc/paper/8275-joint-autoregressive-and-hierarchical-priors-for-learned-image-compression.pdf.

[41] L. Dong et al. "Learning Deep Representations Using Convolutional Auto-Encoders with Symmetric Skip Connections". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Apr. 2018, pp. 3006–3010. DOI: 10.1109/ICASSP.2018.8462085.

[42] Guoping Zhao et al. "Skip-Connected Deep Convolutional Autoencoder for Restoration of Document Images". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018 24th International Conference on Pattern Recognition (ICPR). ISSN: 1051-4651. Aug. 2018, pp. 2935–2940. DOI: 10.1109/ICPR.2018.8546199.

[43] Y. Blau and T. Michaeli. "The Perception-Distortion Tradeoff". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. June 2018, pp. 6228–6237. DOI: 10.1109/CVPR.2018.00652.

[44] Dong Liu, Haochen Zhang, and Zhiwei Xiong. "On The Classification-Distortion-Perception Tradeoff". In: *arXiv:1904.08816 [cs, math]* (Apr. 18, 2019). arXiv: 1904.08816. URL: http://arxiv.org/abs/1904.08816 (visited on 09/21/2020).

[45] Xiyang Luo et al. "The Rate-Distortion-Accuracy Tradeoff: JPEG Case Study". In: *arXiv:2008.00605 [cs, eess]* (Aug. 2, 2020). arXiv: 2008.00605. URL: http://arxiv.org/abs/2008.00605 (visited on 09/21/2020).

[46] Yochai Blau and Tomer Michaeli. "Rethinking Lossy Compression: The Rate-Distortion-Perception Tradeoff". In: *Proceedings of the 36th International Conference on Machine Learning* (July 30, 2019). arXiv: 1901.07821. URL: http://arxiv.org/abs/1901.07821 (visited on 09/21/2020).

[47] Ting Gong et al. “A Comparison of Loss Weighting Strategies for Multi task Learning in Deep Neural Networks”. In: *IEEE Access* 7 (2019), pp. 141627–141632. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2943604. URL: https://www.mendeley.com/catalogue/cb82487d-de66-3fe3-baf7-b4c8fc111688/ (visited on 09/21/2020).

[48] Shikun Liu, Edward Johns, and Andrew J. Davison. “End-To-End Multi-Task Learning With Attention”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[49] Roberto Cipolla, Yarin Gal, and Alex 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*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. ISSN: 2575-7075. June 2018, pp. 7482–7491. DOI: 10.1109/CVPR.2018.00781.

[50] Lukas Liebel and Marco Körner. “Auxiliary Tasks in Multi-task Learning”. In: *arXiv:1805.06334 [cs]* (May 17, 2018). arXiv: 1805.06334. URL: http://arxiv.org/abs/1805.06334 (visited on 09/21/2020).

[51] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 91–99. URL: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf (visited on 10/19/2020).

[52] R. Girshick. “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015 IEEE International Conference on Computer Vision (ICCV). Dec. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[53] Jasper Uijlings et al. “Selective Search for Object Recognition”. In: *International Journal of Computer Vision* 104 (Sept. 1, 2013), pp. 154–171. DOI: 10.1007/s11263-013-0620-5.

[54] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017 IEEE International Conference on Computer Vision (ICCV). ISSN: 2380-7504. Oct. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[55] Kaiming He et al. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”. In: *arXiv:1502.01852 [cs]* (Feb. 6, 2015). arXiv: 1502.01852. URL: http://arxiv.org/abs/1502.01852 (visited on 10/09/2020).

[56] Jarek Duda et al. "The use of asymmetric numeral systems as an accurate replacement for Huffman coding". In: *2015 Picture Coding Symposium (PCS)*. 2015 Picture Coding Symposium (PCS). May 2015, pp. 65–69. DOI: 10.1109/PCS.2015.7170048.

[57] *Kodak image dataset*. URL: http://www.cs.albany.edu/~xypan/research/snr/Kodak.html (visited on 10/16/2020).

[58] Kaichao You et al. *How Does Learning Rate Decay Help Modern Neural Networks?* Sept. 26, 2019. arXiv: 1908.01878 [cs, stat]. URL: http://arxiv.org/abs/1908.01878 (visited on 10/23/2020).

[59] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. "End-to-end optimization of nonlinear transform codes for perceptual quality". In: *2016 Picture Coding Symposium (PCS)*. 2016 Picture Coding Symposium (PCS). ISSN: 2472-7822. Dec. 2016, pp. 1–5. DOI: 10.1109/PCS.2016.7906310.

[60] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *arXiv:1405.0312 [cs]* (Feb. 20, 2015). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312 (visited on 10/10/2020).

[61] Sébastien Marcel and Yann Rodriguez. "Torchvision the machine-vision package of torch". In: *Proceedings of the 18th ACM international conference on Multimedia*. MM '10. New York, NY, USA: Association for Computing Machinery, Oct. 25, 2010, pp. 1485–1488. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874254. URL: https://doi.org/10.1145/1873951.1874254 (visited on 10/06/2020).

[62] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8026–8037. URL: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (visited on 10/16/2020).

[63] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE, June 2016, pp. 3213–3223. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.350. URL: http://ieeexplore.ieee.org/document/7780719/ (visited on 10/10/2020).

[64] Wikipedia contributors. *Pareto efficiency — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Pareto_efficiency&oldid=984808731. [Online; accessed 24-October-2020]. 2020.

[65] *Versatile Video Coding (VVC) reference software VTM-8.2*. GitLab. Available at: https://vcgit.hhi.fraunhofer.de/jvetVVCSoftware_VTM (Accessed on 2020-10-20). (Visited on 10/20/2020).

[66] Frank Brossen et al. "JVET common test conditions and software reference configurations for SDR video". In: *Joint Video Experts Team (JVET), Document: JVET-N1010* (Mar. 2019).

[67] *Portable Network Graphics (PNG)*. URL: http://libpng.org/pub/png/libpng.html (visited on 10/24/2020).

[68] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: International Conference on Learning Representations (ICLR) 2015. Jan. 29, 2017. arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 10/24/2020).

[69] Gisle Bjontegaard. *Calculation of Average PSNR Differences between RD-Curves*. Apr. 2001. URL: https://www.itu.int/wftp3/av-arch/video-site/0104_Aus/VCEG-M33.doc (visited on 10/25/2020).

[70] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.