Exploiting Latent Properties to Optimize Neural Codecs

Anonymous authors
Paper under double-blind review

Abstract

End-to-end image/video codecs are getting competitive compared to traditional compression techniques that have been developed through decades of manual engineering efforts. These trainable codecs have many advantages over traditional techniques such as easy adaptation on perceptual distortion metrics and high performance on specific domains thanks to their learning ability. However, state of the art neural codecs do not take advantage of vector quantization technique and existence of gradient of entropy in decoding device. In this research, we propose some theoretical insights about these two properties (quantization and entropy gradient), and show that this can improve the performances of many off-the-shelf codecs. First, we prove that non-uniform quantization map on neural codec's latent is not necessary. Thus, we improve the performance by using a predefined optimal uniform vector quantization map. Secondly, we theoretically show that gradient of entropy (available at decoder side) is correlated with the gradient of the reconstruction error (which is not available at decoder side). Thus, we use the former as a proxy in order to improve the compression performance. According to our results, we save between 2-4% of rate for the same quality with this proposal, for various pre-trained methods.

1 Introduction

Lossy Image and video compression is a fundamental task in image processing which has become crucial in the time of pandemic and increasing video streaming volume. Thanks to the community's decades long efforts, traditional methods (e.g. VVC) have reached current state of the art rate-distortion (RD) performance and dominate current codecs market. Recently, end-to-end trainable deep models have emerged with promising RD performances by learning the informative latents and modeling the latent distribution. Even though deep learning based models clearly exceed many traditional techniques and surpass human capability for some general computer vision tasks, they are only almost on par with the best traditional image compressing methods when trained to optimized the peak signal-to-noise ratio (PSNR) according to our knowledge.

End-to-end deep compression methods are generally rate-distortion autoencoders (Habibian et al., 2019), where the latents are optimized using a rate-distortion loss function. For perceptual friendly compression, distortion based on a perceptual metric can also be used in the loss function (Blau & Michaeli, 2019). These methods can be seen as a special case of Variational Autoencoder (VAE) models as described in (Kingma & Welling, 2013), where the approximate posterior distribution is a uniform distribution centered on the encoder's outputs (latents) at training time and has a fixed variance output distribution and trainable priors (Theis et al., 2017; Ballé et al., 2017). It was shown that minimizing the evidence lower bound (ELBO) of this special VAE is equivalent to minimizing jointly the mean square error (MSE) of the reconstruction and the entropy of latents w.r.t the priors (Ballé et al., 2018). All proposed models differ mainly by the modelling of priors: using either fully-factorized (Ballé et al., 2017), zero-mean gaussian (Ballé et al., 2018), gaussian (Minnen et al., 2018; Minnen & Singh, 2020) or mixture of gaussian (Cheng et al., 2020), where some methods predict the priors using an autoregressive schema (Minnen et al., 2018; Minnen & Singh, 2020; Cheng et al., 2020; Xie et al., 2021; He et al., 2021) and some improve the priors by global and local context modeling (Qian et al., 2021; Kim et al., 2022). These neural image codecs were extended to the video compression domain by using two VAEs, one for encoding motion information, another one for encoding residual information in end-to-end video compression (Lu et al., 2019; Agustsson et al., 2020; Ladune & Philippe, 2022; Yılmaz & Tekalp, 2021; Pourreza & Cohen, 2021; Li et al., 2021; 2022)

An important step in the building of a neural codec is the quantization of the latent before the entropy coding. All mentioned prior state of the art models use fixed bin-width uniform Scalar Quantization (SQ). Even though Vector Quantization (VQ) is theoretically better (Gersho & Gray, 2012), very limited VQ attempt was done on neural codecs such as in (Agustsson et al., 2017; Zhu et al., 2022). However, VQ does not seem to bring improvement over SQ, which can be explained by various reasons. First, VQ introduces extra trainable parameters to be learned: the codeword centroids. Learning these parameters jointly with all model parameters can introduce additional complexity at training time. Second, since quantization is non-differentiable, a relaxation strategy (Bengio et al., 2013; Agustsson et al., 2017; Zhu et al., 2022) needs to be used at training time, and the continuous relaxation of VQ is less efficient than SQ one.

One of the main principle in general compression is to exploit all information available at the decoder to reconstruct the data. However, even though the gradients of the entropy w.r.t latents are available at decoder side, this information remains unused so far in the literature. Some similar work in the literature have tried to improve the performance of codec during encoding, for example by using specific parameterization (Balcilar et al., 2022), or computationally heavy finetuning solutions (Yang et al., 2020; Guo et al., 2021), either partially (Campos et al., 2019; Lu et al., 2020) or entirely (van Rozendaal et al., 2021). However, all these methods ignore the gradients of the entropy.

In this paper, we propose two original contributions to leverage the properties of the learned latent representation for compression. First, we prove that non-uniform quantization over learned latents is not necessary, even for VQ, contrarily to the usual case. Thus, in order to use the theoretical power of VQ, we propose to apply uniform VQ map over the latents. Since the optimal uniform VQ map is known up to some certain dimensions, we do not need to optimize these quantization centers. This contribution can be applied even without re-training the model if the original model is trained for uniform SQ which is often the case for neural codecs.

Second, we apply Karush–Kuhn–Tucker (KKT) conditions on the neural codec which has not been done to the best of our knowledge. These conditions show that the gradient of the reconstruction error (unavailable at decoder side) is actually correlated with the gradient of entropy (available at decoder side) w.r.t latents. This motivates us to use the available gradient as a proxy for the unavailable gradient in order to improve the performance of the neural codecs without re-training. Our two contributions are generic enough (they do not depend on the encoder-decoder architecture) to bring a rate saving of $2-4\%$ at same quality on several neural codecs architectures.

2 Problem statement and State of the Art

In this section, we recall the basic principles of neural compression, including the training, inference and quantization steps. For an input color image $\mathbf{x} \in \mathbb{R}^{n\times n\times 3}$ to be compressed (the image can be considered square without any loss of generality), the neural codec learns a non-linear encoder $g_a(\mathbf{x};\phi)$, parameterized by the weights ϕ. The output of the encoder, $\mathbf{y} \in \mathbb{R}^{m\times m\times o}$, is called the main embedding (or main latents) of the image. The latent representation is then quantized as $\tilde{\mathbf{y}} = Q(\mathbf{y})$ to obtain the main codes of the image (the dequantization block $\hat{\mathbf{y}} = Q^{-1}(\tilde{\mathbf{y}})$ is used to obtain reconstructed main latents $\hat{\mathbf{y}}$ at decoding side [1]).

The decompressed image $\hat{\mathbf{x}} \in \mathbb{R}^{n\times n\times 3}$ is obtained by the learned deep decoder with $\hat{\mathbf{x}} = g_s(\hat{\mathbf{y}};\theta)$. The neural codec is learned so as to minimize two objectives simultaneously, namely the distortion between $\mathbf{x}$ and $\hat{\mathbf{x}}$, and the length of the bitstream $\tilde{\mathbf{y}}$. The codes are encoded into a bitstream in a lossless manner by any entropy encoder such as range asymmetric numeral systems (RANS) Duda (2009). RANS needs probability mass function (PMF) of each code, and this information is also learned at training time. Since RANS is asymptotically optimal, the lower bound of bitlength according to Shanon's entropy theorem can be used instead of experimental bitlength from RANS in order to make bitlength objective differentiable. Thus, neural codecs use the entropy model to learn the PMF of each codes under determined quantization that allows us to know lower bound of bitlength.

[1] The dequantization stage is sometimes skipped in practice since it can be done inside the first linear operation of the decoder.

Current state of the art neural codecs use the hyperprior entropy model, where the side embedding (or side latents) $\mathbf{z} \in \mathbb{R}^{t \times t \times s}$ is learned by another deep neural network by $\mathbf{z} = h_a(\mathbf{y}; \Phi)$. The side embeddings are quantized as $\tilde{\mathbf{z}} = Q(\mathbf{z})$ to obtain side codes $\tilde{\mathbf{z}}$ followed by dequantization $\hat{\mathbf{z}} = Q^{-1}(\tilde{\mathbf{z}})$ to obtain the reconstructed side latents $\hat{\mathbf{z}}$. The main motivation of side information is to remove any image structure that would be left in the main latent representation y. The hyperprior entropy model first obtains probability density function (PDF) of each scalar latent, assuming it follows a Gaussian distribution where the parameters are obtained by another deep network such as $\boldsymbol{\mu}, \boldsymbol{\sigma} = h_s(\hat{\mathbf{z}}; \Theta)$) [2]. Thus, the hyperprior model's prediction can be defined by $p_{\hat{\mathbf{y}}_{ijk}}(.) := \mathcal{N}(.|\boldsymbol{\mu}_{ijk}, \boldsymbol{\sigma}_{ijk})$. The PMF of latent code $P(\tilde{\mathbf{y}}_{ijk})$ can be written under determined quantization as a function of $p_{\hat{\mathbf{y}}_{ijk}}(.)$. Using this PMFs, the lower bound of the main codes' bitlength can be defined by $-log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)) := -\sum_{ijk} log(P(\tilde{\mathbf{y}}_{ijk}))$. The Factorized entropy model learns the PDF for each $t \times t$ slice of latent defined as $p_{\hat{\mathbf{z}}_{:,:,s}}(.)$. This PDF is enough to have PMF of side codes $P(\hat{\mathbf{z}}_{ijk})$ under determined quantization. Thus, the lower bound of side codes' bitlength can be defined by $-log(p_f(\hat{\mathbf{z}}; \Psi)) := -\sum_{ijk} log(P(\tilde{\mathbf{z}}_{ijk}))$.

In this setting, the deep encoder ($g_a(.; \phi)$), the deep decoder ($g_s(.; \theta)$), the hyperprior entropy $p_h(.; \hat{\mathbf{z}}, \Theta)$ (composed of the deep hyperprior encoder ($h_a(.; \Phi)$), the deep hyperprior decoder ($h_s(.; \Theta)$), and the factorized entropy model $p_f(.; \Psi)$) are the trainable blocks implemented by neural networks. The optimal values of the parameters $\phi, \theta, \Phi, \Theta$ and Ψ are found by minimizing following loss function using the training sample $\mathbf{x}$, comes from distribution of training dataset p_x .

$$\mathcal{L} = \mathop{\mathbb{E}}_{\mathbf{x} \sim p_x} \left[-log(p_f(\hat{\mathbf{z}}; \Psi)) - log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)) + \lambda d(\mathbf{x}, \hat{\mathbf{x}}) \right], \tag{1}$$

where $d(.,.)$ is a distortion measure between the original and the reconstructed image (for example the mean square error). The rate term is sum of the lower bound of bitlength of side information ($-log(p_f(\hat{\mathbf{z}}; \Psi))$) and main information ($-log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta))$). Hyperparameter λ controls the trade-off between the rate (r) and distortion (d) terms.

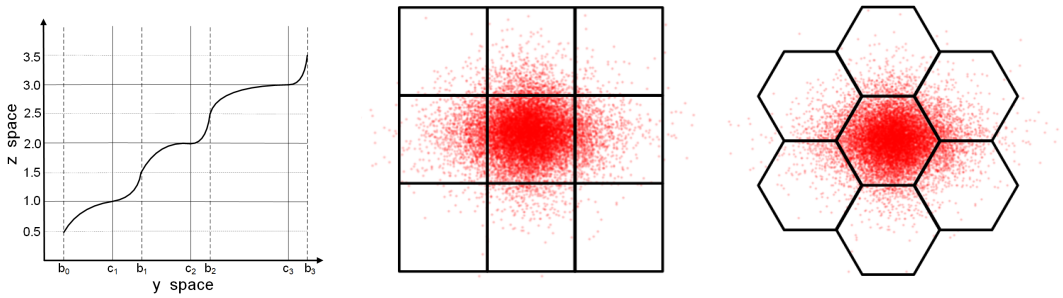
After the training phase, the learned codec can be used at the encoder and decoder side. The image is transformed first to obtain $\tilde{\mathbf{z}}$ and that information is processed by RANS encoder using factorized entropy's PMF table. Then, $\tilde{\mathbf{y}}$ is computed and encoded using RANS with hyperprior entropy's PMF table, obtained by $\tilde{\mathbf{z}}$. At the receiver side, first $\tilde{\mathbf{z}}$ is decoded by RANS decoder using factorized entropy's PMF table. Using $\tilde{\mathbf{z}}$ into the hyperprior model, the PMF table of $\tilde{\mathbf{y}}$ can be computed. Finally, $\tilde{\mathbf{y}}$ is decoded from the bitstream, leading to the reconstructed image using dequantization and deep decoder.

Both quantization step $Q(.)$ and its counterpart dequantization $Q^{-1}(.)$ need to be applied for main and side information. In addition, both factorized and hyperprior entropy models should know the determined quantization technique to determine the used PMF. To the best of our knowledge, all current methods implement quantization as a 1-bin width uniform Scalar Quantization (SQ). This quantization step is implemented by element-wise nearest integer rounding $Q(x) = round(x)$ and its dequantization $Q^{-1}(x) = x$. Thus, the PMF of $\tilde{\mathbf{y}}_{ijk} \in \mathbb{R}$ can be calculated by $P(\tilde{\mathbf{y}}_{ijk}) = \int_{\hat{\mathbf{y}}_{ijk}-0.5}^{\hat{\mathbf{y}}_{ijk}+0.5} p_{\hat{\mathbf{y}}_{ijk}}(x)dx$ where $p_{\hat{\mathbf{y}}_{ijk}}(.)$ is the PDF of latent $\hat{\mathbf{y}}_{ijk}$ learnt by entropy model [3]. Since the nearest integer rounding operation has non-informative gradients, continuous relaxation must be applied at training as $Q(x) = x + \epsilon$, where ϵ is randomly sampled from the uniform distribution $\epsilon \sim U(-0.5, 0.5)$.

However, for the Vector Quantization (VQ) case, latents can be packed into v-dimensional vector $u \in \mathbb{R}^v$ and each u is assigned to a single code. Quantization centers $C_j \in \mathbb{R}^v, j = 1 \dots M$ are learned by entropy model. Quantization step thus amounts to finding nearest center's index as $Q(x) = \arg\min_i ||x - C_i||$, while dequantization returns the quanta center as $Q^{-1}(i) = C_i$ where $i \in \{1 \dots M\}$. In this case, there are M different quantization centers, and M unique codes. Since argmin operator applies hard assignment, it has non-informative gradients and continuous relaxation must also be applied during training. This is generally achieved by softmax operator that assigns all codes to the latent vector with different probabilities, depending

[2] In autoregressive prediction, they are predicted step by step using the previous main latents in a sense of being previous part of the local neighborhood by $\boldsymbol{\mu_i}, \boldsymbol{\sigma_i} = h_s(\hat{\mathbf{z}}, \hat{\mathbf{y}}_{<\mathbf{i}}; \Theta)$

[3] Entropy model can alternatively learn cumulative distribution function CDF, $\sigma_{\hat{\mathbf{y}}_{ijk}}(x)$ instead of PDF and calculate PMF by $P(\tilde{\mathbf{y}}_{ijk}) = \sigma_{\hat{\mathbf{y}}_{ijk}}(\hat{\mathbf{y}}_{ijk} + 0.5) - \sigma_{\hat{\mathbf{y}}_{ijk}}(\hat{\mathbf{y}}_{ijk} - 0.5)$



(a) (b) (c)

Figure 1: a) $f(.)$ transforms non-uniform quantization map to uniform map. b) Uniform SQ grids on 2D c) Optimal uniform VQ grids on 2D.

on the distances to the centers. These probabilities are used by the entropy model to learn the PMF of each quanta center (codes), and also for dequantization which is the expectation of quanta centers under these probabilities during training.

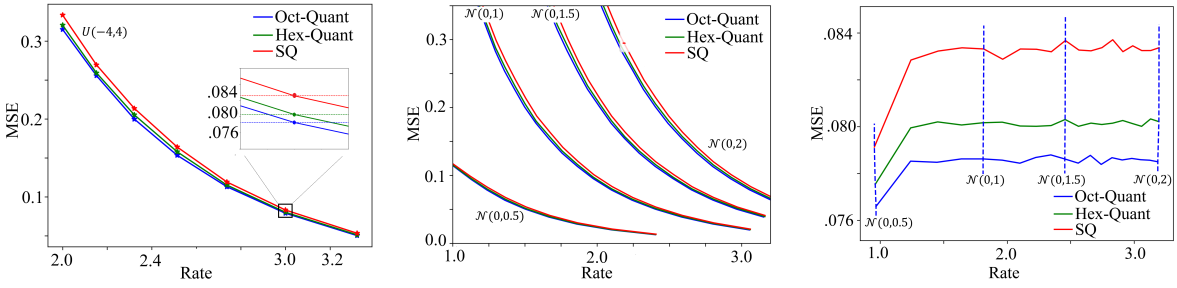
3 Uniform Vector Quantization

Vector quantization (VQ) is theoretically optimal than SQ even if the dimensions are i.i.d (Gersho, 1982), but so far it could not perform significantly better than uniform SQ in neural codecs. The reasons could be that (i) VQ learns non-uniform quantization center over the neural codec's latents which is doubly necessary, (ii) the convergence of these centers might encounter training difficulties and can take longer training time whereas in uniform quantization, the quantization centers are predefined and fixed and (iii) since the dequantization of VQ averages these variable quantization centers in training phase, the reconstructed latents in training phase may be far away from their hard assignment values (values in test phase). This mismatch can be higher than mismatch between nearest integer rounding and its continuous relaxation by additive uniform noise.

Here we improve the performance of vector quantization in neural codec by proposing uniform vector quantization. Our main motivation is to use VQ with uniform grid is based on the sufficiency of uniform SQ among all SQs in the neural codec that we state in the following theorem.

Theorem 1. *If a neural codec has an encoder block* $g_a : \mathbb{R}^{n\times n\times 3} \rightarrow \mathbb{R}^{m\times m\times o}$*, an decoder block* $g_s : \mathbb{R}^{m\times m\times o} \rightarrow \mathbb{R}^{n\times n\times 3}$ *and it needs non-uniform SQ map for the optimal rate-distortion performance, there exists another neural codec that gives the same rate-distortion performance with 1-bin width uniform SQ (nearest integer rounding quantization) whose encoder block is* $f \circ g_a$ *and decoder block is* $g_s \circ f^{-1}$ *where* $f : \mathbb{R}^{m\times m\times o} \rightarrow \mathbb{R}^{m\times m\times o}$ *is an invertible transformation.*

The proof can be found in Appendix A and is based on modelling the one-dimensional quantizer using a memoryless monotonically increasing nonlinearity followed by a uniform fixed-point quantizer as in (Bennett, 1948). We show that an invertible function that can be implemented by neural networks can transform borders and grid centers of the non-uniform quantization map to the uniform quantization map. A simple illustration of this is shown in Figure 1(a). According to Theorem 1 neural codecs with nearest integer rounding quantization is fine among all scalar quantizations as long as it is enough expressive, that its encoder has invertible layers at the bottom of the encoder block and also the inverse of this layers should be at the beginning of the decoder block. Most of the neural codecs do not have invertible blocks, but since the decoder is almost inverse of the encoder (in a sense that it is able to revert back image with negligible distortion) and usually they are enough deep with millions of parameters, we can assume that they are enough expressive. The advantage of using invertible layers between less powerful encoder and decoder block were experimentally found recently in (Shukor et al., 2022). Also, the sufficiency of uniform quantization in neural codecs was discussed in (Ballé et al., 2021). Our theorem verifies these two prior contributions.



(a) (b) (c)

Figure 2: RD performances of different volume uniform SQ, regular hexagon **Hex-Quant** and truncated octahedron **Oct-Quant** grid. a) Uniform source is sampled from $U(-4, 4)$ and zoom-in where the grid volume is unitary. b) Different Gaussian sources. c) RD plot of unitary volume grids for Gaussian sources.

In the following subsection we make connection between the space tessellation and uniform VQ.

3.1 Space tessellation grids

We propose uniform vector quantization based on the space tessellation in the latent space of neural codes, where the predefined shape covers the entire high dimensional space. Thus, the quantization grids are fixed instead of learning it. The nearest integer quantization grid in 1D is equivalent to unit square grid in 2D, as shown in Figure 1(b) for zero-mean Gaussian source, however it might not be the best partition of the space with uniform grids, thus using nearest integer quantization (i.e square grids in 2D) directly in the neural codec might not yield better RD performance. Following remark shows the existence of entropy constrained optimal space tesellation and can be used for uniform vector quantization.

Remark 1. *VQ constraints with uniform grid produces the optimal space tessellation which refers to a pattern of v-dimensional shapes that fit perfectly together without any gaps and have minimum inertia. The optimal shapes are regular hexagon for 2D, truncated octohedron for 3D and polytope $\mathbf{D}_4$ in 4D case (Gersho, 1979).*

The above remark shows that instead of learning the grids centre and optimal partition, we can use the optimal space tessellation grids at the corresponding dimensions. In this paper our proposal for 2D space is to use regular hexagon grid that has unitary volume as shown in Figure 1(c). Theoretical advantage of space tessellation grid compared to the nearest integer quantization is shown for uniform distribution in Appendix B. We also represent the simulation results under uniform and Gaussian sources in Figure 2a-b. Figure 2c shows RD performance of unitary grids under different zero-mean Gaussian source is particularly important. Because the main information in modern neural codec is zero mean different scale Gaussian distribution and quantized with unitary grids. Figure 2c shows that when the scale parameter of Gaussian increases, it approximates the theoretical known RD performance of uniform source. Now we show that how our proposed uniform vector quantization can be applied in the off-the-shelf neural codec without re-training.

3.2 Uniform VQ with off-the-self neural codec

Let $\mathbf{y} \in \mathbb{R}^{m \times m \times o}$ be the latents of the deep encoder, v be the dimension where the optimal space tessellation is performed, $\mathbf{c}^{(i)} \in \mathbb{R}^v, i = 1 \dots M$ be the M grid centers of the v-dimensional shape and $\mathbf{c}_j^{(i)} \in \mathbb{R}$ is the scalar value on j-th dimension of the center $\mathbf{c}^{(i)}$. In order to quantize the latents, we reshape the latents $\mathbf{y}$ into pseudo v-dimensional latents such that $\mathbf{y} \in \mathbb{R}^{m \times m \times o} \rightarrow \check{\mathbf{y}} \in \mathbb{R}^{b \times v}$, where $b = \frac{m.n.o}{v}$. The quantization is performed by assigning the nearest neighbours grid centers, as $\tilde{\mathbf{y}}_j = Q(\check{\mathbf{y}}_j) = \arg\min_i ||\check{\mathbf{y}}_j - \mathbf{c}^{(i)}||$, where $\tilde{\mathbf{y}}_j \in \{1 \dots M\}, j = 1 \dots b$ are the codes to be encoded into bitstream. In practice, instead of reshaping all latents together into v dimension, we can reshape the latents whose learned PDFs are the same (i.e coming from the same distribution). This approaches does not have significantly different results but needs smaller PMF table to be kept in decoding device.

To encode the codes ($\tilde{\mathbf{y}}$) into bit-stream, we cannot use the off-the-shelf neural codec's 1D entropy model's PMF calculation, as our latents are v-dimensional latents and domain is different. To this end, we propose to create the PMF by integrating the pseudo-v dimensional PDF ($\prod_{j=1}^{v} p_{\mathbf{c}_j^{(i)}}(u_j)$) inside the grid G whose center is located on $\mathbf{c}^{(i)}$:

$$P(\mathbf{c}^{(i)}) = \int_{G+\mathbf{c}^{(i)}} \left(\prod_{j=1}^{v} p_{\mathbf{c}_j^{(i)}}(u_j) \right) du \tag{2}$$

Here $P(\mathbf{c}^{(i)})$ is the PMF of i-th symbol in the dictionary and $p_{\mathbf{c}_j^{(i)}} : \mathbb{R} \to \mathbb{R}$ is PDF of corresponding latent, learned by entropy model in the baseline model. Since there is no closed form solution of integral of multi dimensional Gaussian distribution over hexagonal and truncated octahedron domain (Savaux & Le Magoarou, 2020), we use numeric solvers to approximate the solution of equation 2. We can reach numeric solution for main codes PMFs where $p_{\mathbf{c}_j^{(i)}}(.)$ is the Gaussian distribution and also non-parametric $p_{\mathbf{c}_j^{(i)}}(.)$ for side code's PMF. In Appendix E, we show how to calculate equation 2 for hexagonal domain. Another way to calculate those integral by monte-carlo simulation for known PDFs $p_{\mathbf{c}_j^{(i)}}(.)$ before deployment which we use it for truncated octahedron domain.

4 Forgotten Information: Gradient of Entropy

In this section, we describe how the gradients of the entropy in the receiver side is used to improve the reconstruction performance. When receiver decodes side latent, it can also know the gradients of side entropy w.r.t side latent. After decoding main latent, it can also compute the gradients of main entropy w.r.t main latent. Since these gradients are only available after decoding the latent codes, they seem not useful in the first sight. This could be the reason why so far these gradients are never used in test phase. Here, we propose to use the gradients through the analysis of Karush-Kuhn-Tucker conditions, and we claim that first gradient is correlated with gradient of main entropy w.r.t side latent and the second one is correlated with gradient of reconstruction error w.r.t main latent. Thus, using the first one as in its correlated gradient, we can decrease the bitlength of main information and using the second one as in its correlated gradient, we can decrease the reconstruction error.

We can see the neural codec's loss function in equation 1 as an unconstrained multi-objectives optimization problem, where the objectives are minimum bitlength of side information, minimum bitlength of main information and minimum reconstruction error. The optimal solution of multi-objective problem is called Pareto Optimal which is a solution where no objective can be made better off without making at least one objective worse off (Miettinen, 2012). The following remark shows a useful property of a solution of unconstrained multi-objective optimization problems.

Remark 2. *A solution of the multi-objective optimization problem is Pareto Optimal, if and only if it satisfies Karush–Kuhn–Tucker (KKT) conditions. More specifically, in unconstrained multi-objective optimization problems case, if the aim of the problem is* $w^* = \arg\min_w(\sum_i \alpha_i.\mathcal{L}_i(w))$*; where* $\alpha_i \geq 0$*,* $\sum \alpha_i = 1$ *and* $\mathcal{L}_i$ *is the* i*-th objective to be minimized, the solution* w^* *is Pareto Optimal if and only if it satisfies the following (Désidéri, 2012).*

$$\sum_i \alpha_i \nabla_w \mathcal{L}_i(w^*)) = 0.$$

In plain words, this shows that on the optimal solution point, all forces driven by gradients cancel each other out and the solution reaches saddle point. This property is used to test the optimality of the candidate solutions. This remark is also valid for end-to-end compression models. Following theorem shows how to use KKT conditions for the end-to-end image compression models.

Theorem 2. *An end-to-end compression model optimized with* λ *trade-off is Pareto Optimal, if and only if the following two conditions are met.*

$$\mathbb{E}_{\mathbf{x}\sim p_x}\left[\nabla_{\hat{\mathbf{z}}}(-log(p_f(\hat{\mathbf{z}};\Psi))) + \nabla_{\hat{\mathbf{z}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))\right] = 0 \tag{3}$$

$$\mathbb{E}_{\mathbf{x}\sim p_x}\left[\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta))) + \lambda\nabla_{\hat{\mathbf{y}}}(d(\mathbf{x}, g_s(\hat{\mathbf{y}};\theta)))\right] = 0 \tag{4}$$

The proof can be found in Appendix C. It is straight-forward, but the result of this theorem is significant. One can interpret the theorem that if the existed end-to-end models is optimal (atleast in terms of training performance, not in terms of compressing performance), gradient of side information's entropy w.r.t side latents $\nabla_{\hat{\mathbf{z}}}(-log(p_f(\hat{\mathbf{z}};\Psi)))$ and gradient of main information's entropy w.r.t side latents $\nabla_{\hat{\mathbf{z}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))$ cancel themselves out in expectation. It is the same for the second condition as well. We can claim that main information's entropy w.r.t main latents $\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))$ and weighted gradient of reconstruction error w.r.t main latents $\lambda\nabla_{\hat{\mathbf{y}}}(d(\mathbf{x},g_s(\hat{\mathbf{y}};\theta)))$ cancels themselves out in expectation.

If these conditions were valid for every single image, given test image's available gradient would be negative of unavailable gradient, thus they would have -1 correlation coefficient. However, these conditions are valid in average of images in train set and cannot guarantee the correlation for every single test image. On the other hand, we can still assume that since the train set somehow represents the test image, at least these two gradients are correlated in test image, even though they are not exact negative of themselves. This assumption is validated for different neural codecs with different test set, and presented in Appendix D and section 5. The scatter plot of a sample image's gradient of main latents and its correlated one are shown on Figure 3. These gradients were calculated on a neural codec described in Minnen et al. (2018) and implemented in Bégaint et al. (2020). According to our test on widespread neural codecs, we have found quite strong correlation on main latent's gradient (between -0.1 to -0.5), but weak correlation (between -0.15 to 0.1) on side latent's gradient.

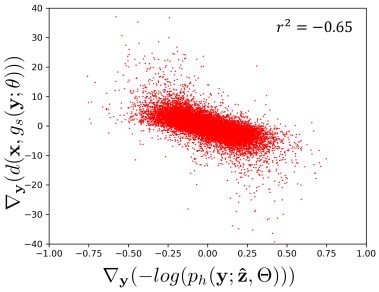


Figure 3: Correlation between gradients of the entropy and the reconstruction error with respect to the main latents before quantization.

Latent Shift wrt Gradients. By definition of gradient based optimization, $\hat{\mathbf{z}}$ needs to take a step in negative direction of $\nabla_{\hat{\mathbf{z}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))$ in order to decrease main information bitlength $-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)$. However $\nabla_{\hat{\mathbf{z}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))$ is not available before decoding $\hat{\mathbf{y}}$ in decoding device, but at least the correlated gradient $\nabla_{\hat{\mathbf{z}}}(-log(p_f(\hat{\mathbf{z}};\Psi)))$ is known after decoding $\hat{\mathbf{z}}$. We claim that there is a real number step size ρ_f^* that decrease the bitlength of main information such that

$$-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta) \geq -log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}}+\rho_f^*\nabla_{\hat{\mathbf{z}}}(-log(p_f(\hat{\mathbf{z}};\Psi)))),\Theta),$$

ρ_f^* can be obtained by brutal force out of handful candidates or any optimization method to find optimal one such that;

$$\rho_f^* = \arg\min_{\rho_f}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}}+\rho_f\nabla_{\hat{\mathbf{z}}}(-log(p_f(\hat{\mathbf{z}};\Psi)))),\Theta)). \tag{5}$$

For the second condition, $\hat{\mathbf{y}}$ needs to take a step in negative direction of $\nabla_{\hat{\mathbf{y}}}(d(\mathbf{x},g_s(\hat{\mathbf{y}};\theta)))$ in order to decrease reconstruction error $d(\mathbf{x},g_s(\hat{\mathbf{y}};\theta))$. Since $\nabla_{\hat{\mathbf{y}}}(d(\mathbf{x},g_s(\hat{\mathbf{y}};\theta)))$ is never available in decoding device, but at least the correlated gradient $\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))$ is known after decoding $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$. We claim that there is a real number step size ρ_h^* that decrease the reconstruction error such that

$$d(\mathbf{x},g_s(\hat{\mathbf{y}};\theta)) \geq d(\mathbf{x},g_s(\hat{\mathbf{y}}+\rho_h^*\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)));\theta)),$$

ρ_h^* can be found by brute-force search out of handful candidates or any optimization method to find optimal one such that;

$$\rho_h^* = \arg\min_{\rho_h}(d(\mathbf{x},g_s(\hat{\mathbf{y}}+\rho_h\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)));\theta))). \tag{6}$$

In summary, our proposal can be seen as shifting the side latent by $\hat{\mathbf{z}} \leftarrow \hat{\mathbf{z}}+\rho_f^*\nabla_{\hat{\mathbf{z}}}(-log(p_f(\hat{\mathbf{z}};\Psi)))$ after decoding $\hat{\mathbf{z}}$ and shifting the main latent by $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}}+\rho_h^*\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}};\hat{\mathbf{z}},\Theta)))$ after decoding $\hat{\mathbf{y}}$ while the best step sizes $\rho_f^*,\rho_h^* \in \mathbb{R}$ are to be found in encoding time and added to the bitstream explicitly with fixed bitlength. It is noted that during the encoding, the main latents $\hat{\mathbf{y}}$ are encoded wrt entropy model predicted by the shifted side latent. We show that the shifted side latents shortens the main information's bitlength and shifted main latents results in the better reconstruction performance.

5 Experimental Results

We use CompressAI library (Bégaint et al., 2020) to test our contributions on 4 pre-trained neural image codecs named **bmshj2018-factorized** in Ballé et al. (2017), **mbt2018-mean** and **mbt2018** in Minnen et al. (2018), **cheng2020-attn** in Cheng et al. (2020), and 2 additional pre-trained codec named **invcompress** in Xie et al. (2021) and **DCVC-intra** in Li et al. (2022) with all intra mode and two neural video codec named **SSF** in Agustsson et al. (2020) and **DCVC** in Li et al. (2022). For the evaluation, we use Kodak dataset (Kodak) and Clic-2021 Challenge's Professional dataset (CLI). The codecs are taken off the shelf and are not retrained. The rate is calculated from the final length of the compressed data and RGB PSNR is used for distortion. We asses the performance using the bd-rate (Bjontegaard, 2001) on the rate range used in compressAI: 0.1bpp to 1.6bpp for image codecs and 0.03bpp to 0.35bpp for video codecs. We report our uniform VQ proposals under the name of **Hex-Quant** for regular hexagonal. We denote by **Latent Shift** the gradient based improvement and **Join** the results using both methods.

Table 1: Average BD-Rate gains of our proposals for different baseline image codecs on 2 image datasets.

	Kodak Test Set			Clic-2021 Test Set		
Baseline Codec	Hex-Quant.	Latent Shift	Join	Hex-Quant.	Latent Shift	Join
bmshj2018-factorized	-0.62%	-0.49%	-1.08%	-1.78%	-0.69%	-2.26%
mbt2018-mean	-0.88%	-1.27%	-2.24%	-0.76%	-1.21%	-2.03%
mbt2018	-0.55%	-1.44%	-1.98%	-0.66%	-1.71%	-2.33%
cheng2020-attn	-0.16%	-0.46%	-0.73%	-0.32%	-0.72%	-1.15%
InvCompress	-0.21%	-0.55%	-0.82%	-0.52%	-0.63%	-1.12%
DCVC-intra mode	-0.97%	-0.30%	-1.28%	-1.22%	-0.11%	-1.44%

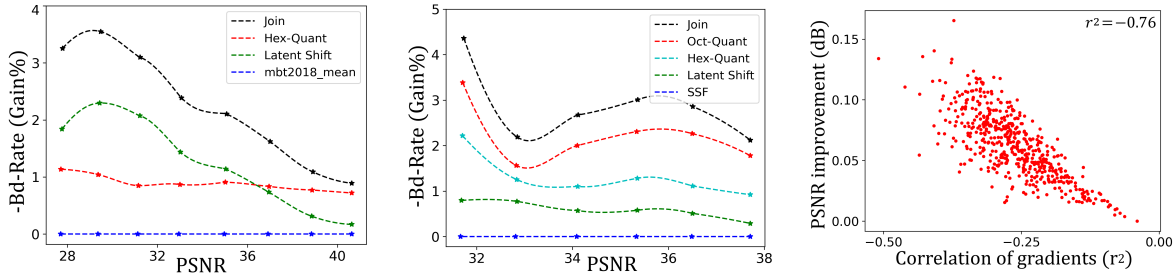
Results on Image Codecs: In Table 1, we show the results of the methods for different datasets and codecs. Since our two proposals are orthogonal to each other, the gain of the joined methods is almost the sum of the two. However, after uniform VQ, the reconstructed latents change compare to uniform SQ, thus, gradients wrt this latents change as well. This variation creates slightly different results compared to **Latent Shift** only, thus **Join** results are not the exact sum. Since uniform VQ's reconstructed latent is much more closer to the latents before quantization, nearly in all cases, **Latent Shift** with uniform VQ gives better contribution than only **Latent Shift**.

The bd-rate (gain in %) compared to **mbt2018-mean** for different quality is shown in Figure 4a on the Kodak dataset. Even though our proposal has a 2.24% gain, the gain on lower quality (thus lower rate) is bigger (3.5%). More detailed results of the gain w.r.t. to reconstruction quality are shown in Appendix G. According to our tests, since the side latent's correlation's is weaker, the most important gain of **Latent Shift** comes from the gradient of the main latent. The analysis of the source of the gain of **Latent Shift** and its performance against some alternatives are presented in Appendix D.

Table 2: Average BD-Rate of our proposals for SSF and DCVC video codecs on UVG dataset and Bunny video in low-delay configuration. **Join** refers using **Oct-Quant** and **Latent Shift**.

	SSF				DCVC			
Video	Hex-Quant.	Oct-Quant.	Latent Shift	Join	Hex-Quant.	Oct-Quant.	Latent Shift	Join
Beauty	-1.31%	-1.60%	-0.13%	-1.76%	-1.92%	-1.97%	-0.10%	-2.02%
Bosphorus	-1.88%	-2.29%	-0.70%	-3.23%	-1.23%	-1.48%	-0.62%	-2.05%
Honeybee	-1.75%	-1.10%	-0.88%	-2.00%	-1.80%	-2.13%	-0.56%	-2.65%
Jockey	-1.04%	-1.48%	-0.06%	-1.57%	-1.42%	-1.32%	-0.11%	-1.51%
UVG Average	-1.31%	-1.99%	-0.60%	-2.70%	-1.52%	-1.64%	-0.41%	-2.05%
ReadySteadGo	-1.03%	-2.00%	-1.52%	-3.74%	-1.29%	-1.30%	-1.08%	-2.35%
ShakeNDry	-1.38%	-2.83%	-0.35%	-3.25%	-1.55%	-1.65%	-0.23%	-1.92%
YatchRide	-1.03%	-1.67%	-0.14%	-1.87%	-1.42%	-1.61%	-0.18%	-1.83%
Bunny	-1.12%	-1.87%	-1.31%	-3.24%	-0.90%	-1.02%	-1.21%	-2.25%

Results on Video Codec: In Table 2, we show the results of the **SSF** and **DCVC** video codecs for different sequences. The **SSF** (Agustsson et al., 2020) was proposed for low-delay mode: frames are coded sequentially using the previously reconstructed frame and an intra frame is inserted every 8 frames (UVG videos are divided into 75 Intra periods). I frame encoding uses the same VAE method, so our implementation on I frame is the same as the one described above. However, each P frame consists of motion information and residual information encoded by two different VAE models. Since the motion information represents only a fraction of the total bitrate $(3-4\%)$, we apply our proposal on the residual coding only. When we apply **Oct-Quant** and **Latent Shift** together, it gains $2-2.7\%$ of bitrate for the same quality on UVG dataset on average. Again, the gain at lower rate is larger (more than 4%) as shown in Figure 4b. More analysis on quantization effects can be found in Appendix D.



(a) (b) (c)

Figure 4: BD-Rates of our proposals from baseline codecs for different quality. a) mbt2018mean image codec on Kodak test set b) SSF video codec on UVG test set. c) Correlation between improvement on reconstruction quality and correlation of gradients on Clic and Kodak datasets.

Gradients correlation: The **Latent Shift**'s gain depends on how the gradients are correlated. To assess this correlation, we show the scatter plot between **Latent Shift**'s individual gains in dB and actual correlation coefficient between gradients for all test images and for all quality level in Figure 4c. We found a correlation of -0.76 where this correlation is independent from datasets, reconstruction quality or model. Since the neural video codec was optimized for the loss of all frames in a GOP, the gradients' sum in equation 3 and equation 4 are zero in expectation for both the frames in the GOP and over the datasets. This smooths the sum of gradients even further and decrease the correlations compare to the image codecs. It explains why the the correlation of the gradients are lower and also why **Latent Shift**'s individual gain is lower in video codecs.

Complexity analysis: We have evaluated the complexity of our proposed over the baseline and the detailed analysis are presented in Appendix F. Our uniform VQ introduces about 3-5% additional complexity during encoding and 2% during decoding, and the **Latent Shift** introduces about 5-10 times additional complexity during encoding and about 1% during decoding over the baseline method.

6 Conclusion

In this work, we have proposed two orthogonal methods to improve further the latent representation of generic compressive variational auto-encoders (VAE). Firstly, we exploit further the remaining redundancy in the latent during the quantization stage. To do so, we demonstrate that a uniform VQ method improves a VAE trained using with uniform scalar quantization. Secondly, we exploit the correlation of the gradient of the entropy and the reconstruction error to improve the latent representation. The combination of these two methods improve the latent representation, and bring significant gains on top of several state-of-the-art compressive auto-encoders, without any need of retraining. From these results, several improvements can be foreseen. First, the correlation of the gradients depends on the training set according to the definition of the KKT conditions. However, even though different models use the same training set and training procedure, their gradients' correlation coefficient may be different. Thus, it might be interesting to explore

the connection of certain type of model architectures with the correlation of the gradients. Secondly, the uniform VQ process was applied without retraining. Even though VQ quantization error is lower on average, the maximum quantization error can be higher. Since the decoder block cannot see that individual high errors during the training, the model becomes sub-optimal. This can be solved by fine-tuning the decoder block or the entire model using a continuous relaxation of the uniform VQ grid.

References

CLIC: Challenge on learned image compression. `http://compression.cc`(Cited on 8)

E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. IEEE Transactions on Information Theory, 48(8):2201–2214, 2002. doi: 10.1109/TIT.2002.800499. (Cited on 20)

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. Advances in neural information processing systems, 30, 2017. (Cited on 2)

Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8503–8512, 2020. (Cited on 1, 8, 9)

Muhammet Balcilar, Bharath Damodaran, and Pierre Hellier. Reducing the amortization gap of entropy bottleneck in end-to-end image compression. In Picture Coding Symposium (PCS), 2022. (Cited on 2)

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In ICLR, 2017. (Cited on 1, 8, 20)

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In ICLR, 2018. (Cited on 1)

Johannes Ballé, Philip A. Chou, David Minnen, Saurabh Singh, Nick Johnston, Eirikur Agustsson, Sung Jin Hwang, and George Toderici. Nonlinear transform coding. IEEE J. Sel. Top. Signal Process., 15(2):339–353, 2021. doi: 10.1109/JSTSP.2020.3034501. URL `https://doi.org/10.1109/JSTSP.2020.3034501`. (Cited on 4)

Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. arXiv preprint arXiv:2011.03029, 2020. (Cited on 7, 8)

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432, 2013. (Cited on 2)

William Ralph Bennett. Spectra of quantized signals. The Bell System Technical Journal, 27(3):446–472, 1948. (Cited on 4)

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. VCEG-M33, 2001. (Cited on 8, 16)

Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pp. 675–685. PMLR, 09–15 Jun 2019. (Cited on 1)

Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers. Content adaptive optimization for neural image compression. In CVPR Workshops, June 2019. (Cited on 2)

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In CVPR, 2020. (Cited on 1, 8, 20)

J. Conway and N. Sloane. Fast quantizing and decoding and algorithms for lattice quantizers and codes. IEEE Transactions on Information Theory, 28(2):227–232, 1982. doi: 10.1109/TIT.1982.1056484. (Cited on 20)

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. Comptes Rendus Mathematique, 350(5-6):313–318, 2012. (Cited on 6)

Jarek Duda. Asymmetric numeral systems. arXiv preprint arXiv:0902.0271, 2009. (Cited on 2)

Allen Gersho. Asymptotically optimal block quantization. IEEE Transactions on information theory, 25(4): 373–380, 1979. (Cited on 5)

Allen Gersho. On the structure of vector quantizers. IEEE Transactions on Information Theory, 28(2): 157–166, 1982. doi: 10.1109/TIT.1982.1056457. (Cited on 4)

Allen Gersho and Robert M Gray. Vector quantization and signal compression, volume 159. Springer Science & Business Media, 2012. (Cited on 2)

Zongyu Guo, Zhizheng Zhang, Runsen Feng, and Zhibo Chen. Soft then hard: Rethinking the quantization in neural image compression. In Marina Meila and Tong Zhang (eds.), Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 3920–3929. PMLR, 18–24 Jul 2021. (Cited on 2)

Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7033–7042, 2019. (Cited on 1)

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14771–14780, 2021. (Cited on 1)

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989. (Cited on 13)

Jun-Hyuk Kim, Byeongho Heo, and Jong-Seok Lee. Joint global and local hierarchical priors for learned image compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5992–6001, June 2022. (Cited on 1)

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In ICLR, 2013. (Cited on 1)

Eastman Kodak. Kodak Lossless True Color Image Suite (PhotoCD PCD0992). URL `http://r0k.us/graphics/kodak`. (Cited on 8)

Théo Ladune and Pierrick Philippe. Aivc: Artificial intelligence based video codec. arXiv preprint arXiv:2202.04365, 2022. (Cited on 1)

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), Advances in Neural Information Processing Systems, volume 34, pp. 18114–18125. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/96b250a90d3cf0868c83f8c965142d2a-Paper.pdf`. (Cited on 1)

Jiahao Li, Bin Li, and Yan Lu. Hybrid spatial-temporal entropy modelling for neural video compression. In Proceedings of the 30th ACM International Conference on Multimedia, MM '22, pp. 1503–1511, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392037. doi: 10.1145/3503161.3547845. URL `https://doi.org/10.1145/3503161.3547845`. (Cited on 1, 8)

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11006–11015, 2019. (Cited on 1)

Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In ECCV, volume 12347, pp. 456–472, 2020. (Cited on 2)

Kaisa Miettinen. Nonlinear multiobjective optimization, volume 12. Springer Science & Business Media, 2012. (Cited on 6)

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In ICIP, pp. 3339–3343, 2020. (Cited on 1)

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 31, 2018. (Cited on 1, 7, 8, 17, 20)

Reza Pourreza and Taco Cohen. Extending neural p-frame codecs for b-frame coding. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6680–6689, 2021. (Cited on 1)

Yichen Qian, Zhiyu Tan, Xiuyu Sun, Ming Lin, Dongyang Li, Zhenhong Sun, Hao Li, and Rong Jin. Learning accurate entropy model with global reference for image compression. In Proceedings of the International Conference on Learning Representations (ICLR), 2021. (Cited on 1)

Vincent Savaux and Luc Le Magoarou. On the computation of integrals of bivariate gaussian distribution. In 2020 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6. IEEE, 2020. (Cited on 6, 17)

Mustafa Shukor, Bharath Bushan Damodaran, Xu Yao, and Pierre Hellier. Video coding using learned latent gan compression. In Proceedings of the 30th ACM International Conference on Multimedia (MM '22), 2022. (Cited on 4)

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In ICLR, 2017. (Cited on 1)

Ties van Rozendaal, Iris AM Huijben, and Taco S Cohen. Overfitting for fun and profit: Instance-adaptive data compression. In ICLR, 2021. (Cited on 2)

Yueqi Xie, Ka Leong Cheng, and Qifeng Chen. Enhanced invertible encoding for learned image compression. In Proceedings of the ACM International Conference on Multimedia, 2021. (Cited on 1, 8)

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 573–584, 2020. (Cited on 2)

M Akın Yılmaz and A Murat Tekalp. End-to-end rate-distortion optimized learned hierarchical bi-directional video compression. IEEE Transactions on Image Processing, 2021. (Cited on 1)

Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, and Heng Tao Shen. Unified multivariate gaussian mixture for efficient neural image compression. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 17591–17600, 2022. doi: 10.1109/CVPR52688.2022.01709. (Cited on 2)

A Proof of Theorem 1

Proof. Scalar quantization map can be see as $n+1$ border values with $b_0 < b_1 < \cdots < b_n$ and n quantization center with $c_1 < c_2 < \cdots < c_n$. Any latent y with $b_{i-1} \leq y < b_i$ should be quantized to the point c_i. Quantization map can be represented by an ordered set consisting of all borders and centers such as $\mathbb{M} = \{b_0 < c_1 < b_1 < c_2 < b_2 < \cdots < c_n < b_n\}$ where $|\mathbb{M}| = 2n+1$. When the quantization map is non-uniform, the difference between consecutive elements in the set are not necessarily equal, i.e. $\exists(i,j), \mathbb{M}_i - \mathbb{M}_{i-1} \neq \mathbb{M}_j - \mathbb{M}_{j-1}$. Nearest integer quantization map can be denoted as $\mathbb{M}^{(u)} = \{0.5, 1, 1.5, 2, 2.5, \ldots n, n+0.5\}$ or simply $\mathbb{M}_i^{(u)} = 0.5i$ thus,$\forall i, \mathbb{M}_i^{(u)} - \mathbb{M}_{i-1}^{(u)} = 0.5$. Let's assume than any arbitrary $\mathbb{M}$ is the optimal scalar quantization map of a neural codec's latent y obtained by $y = g_a(x)$. Since $\mathbb{M}$ and $\mathbb{M}^{(u)}$ are both monotonic increasing set, there exists a bijective function $f(.)$ that maps $\forall i, \mathbb{M}_i$ to $\mathbb{M}_i^{(u)}$, and $f^{-1}(.)$ maps $M_i^{(u)}$ to $\mathbb{M}_i$, $\forall i$. According to universality theorem Hornik et al. (1989), the function $f(.)$ can be implemented by a multi layer neural network.

Two codecs' performances are equal if and only if entropy of their latents are equal and their reconstructions are the same. First, we start by showing that their entropy's are the same by showing that the corresponding center's PMF are equal in both space, i.e. $\forall i, P(i) = P^{(u)}(i)$. We can write the i-th quantization center's PMF as $P(i) = \int_{b_{i-1}}^{b_i} p(y)dy$ where y is a point in g_a's output space. Any point y can be transformed to a new space by $z = f(y)$. We can write the i-th quantization center's PMF as $P^{(u)}(i) = \int_{i-0.5}^{i+0.5} p(z)dz$ in this space. We can rewrite it as $P^{(u)}(i) = \int_{f^{-1}(i-0.5)}^{f^{-1}(i+0.5)} p(f^{-1}(z))df^{-1}(z)$. Since $f^{-1}(i+0.5) = b_i$, $f^{-1}(i-0.5) = b_{i-1}$ and $f^{-1}(z) = y$, we can write $\forall i, P^{(u)}(i) = \int_{b_{i-1}}^{b_i} p(y)dy = P(i)$.

The output of a deep decoder is $g_s(c_i)$ if the latent $y = g_a(x)$ meets $b_{i-1} \leq y < b_i$. Any latent $b_{i-1} \leq y < b_i$ is mapped to $f(b_{i-1}) \leq f(y) < f(b_i)$ thus $i - 0.5 \leq z < i + 0.5$. The latent z which lies between $i-0.5$ to $i+0.5$, can be quantized to i in the new space. Since the decoder applies $g_s(f^{-1}(z))$, its output should be $g_s(f^{-1}(i))$. Since $f^{-1}(i) = c_i$, it gives $g_s(c_i)$. □

B Advantage of Space Tessellation Grid on Quantization

When the source has a uniform distribution, quantization using a truncated octahedron gives better RD performance compared to regular hexagonal grids, and regular hexagonal grid gives better RD performance than uniform SQ grids (nearest integer rounding). In order to show the superiority of a method over another in terms of RD performance, it is enough to compare the reconstruction error at equal bit-rate. Since the equal volume grids have the same probability under uniform distribution, the rate is equal for the three cases. Since the distributions are identical, we just need to compute the mean square error for each types of grid at some position, for example the origin for 1D, 2D and 3D cases respectively.

MSE of uniform SQ grid can be written as the integral of square error normalized by the grid size:

$$MSE^{(s)}(u) = \frac{1}{u} \int_{-u/2}^{u/2} x^2 dx. \tag{7}$$

It gives $MSE^{(s)}(u) = u^2/12 \approx 0.0833u^2$.

Hexagonal grid case: For hexagonal grid, we need to double integrate over the hexagonal domain. Figure 5a shows an hexagon with a side a length a located at the origin. We also show each side's functions in 2D space. We divide the hexagon into two parts for positive and negative y and calculate the analytic integral for these two region separately. We then normalize the sum of squares by area of the area of the hexagon, which is $3\sqrt{3}a^2/2$.

$$MSE^{(h)}(a) = \frac{2}{3\sqrt{3}a^2} \left(\int_0^{a\frac{\sqrt{3}}{2}} \int_{-a+\frac{y}{\sqrt{3}}}^{a-\frac{y}{\sqrt{3}}} \frac{x^2+y^2}{2} dxdy + \int_{-a\frac{\sqrt{3}}{2}}^{0} \int_{-a-\frac{y}{\sqrt{3}}}^{a+\frac{y}{\sqrt{3}}} \frac{x^2+y^2}{2} dxdy \right). \tag{8}$$

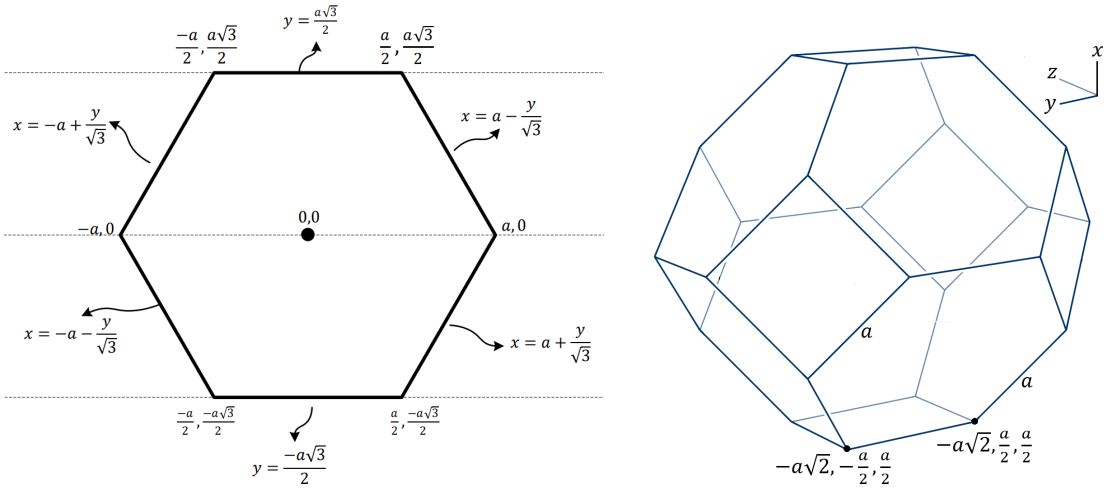


Figure 5: a) Regular hexagon and b) Truncated octahedron located at origin.

When we first integrate the positive part of integral over x between $-a + \frac{y}{\sqrt{3}}$ and $a - \frac{y}{\sqrt{3}}$, followed by the integral over y between 0 and $a\frac{\sqrt{3}}{2}$. The integral for the positive region is is then $\frac{5\sqrt{3}a^4}{32}$. The same can be done for the negative region, but since the distribution is uniform, it gives the exact same result, thus we conclude that $MSE^{(h)}(a) = \frac{5a^2}{24}$. In order to obtain the same volume for the grid, hexagon's area should be u^2. Since the hexagon's area is $3\sqrt{3}a^2/2$ for a side length if a, we find that $a = \frac{\sqrt{2}u}{\sqrt{3\sqrt{3}}}$. Thus $MSE^{(h)}(u) = \frac{5\sqrt{3}u^2}{108} \approx 0.0801u^2$.

Truncated octahedron case: For the MSE of the truncated octahedron grid, we need to integrate the mean square error of each 3 dimension over the truncated octahedron domain. In figure 5b, a truncated octahedron with a side length of a is shown. The solution can be obtained by integrating one octant, multiply it by 8 and normalize by its volume which is $8\sqrt{2}a^3$ as follows:

$$MSE^{(o)}(a) = \frac{8}{8\sqrt{2}a^3} \int_0^{\sqrt{2}a} \int_0^{min(\sqrt{2}a, 3\sqrt{2}a/2 - x)} \int_0^{min(\sqrt{2}a, 3\sqrt{2}a/2 - x - y)} \left(\frac{x^2 + y^2 + z^2}{3} \right) dz dy dx \tag{9}$$

First, we integrate over z, y and then x, which gives the solution $MSE^{(o)}(a) = \frac{19}{48}a^2$. To compare with the same grid volume, the truncated octahedron should have a volume of u^3. Since the volume is $8\sqrt{2}a^3$ when one side length is a, we can find that $a = \frac{u}{(8\sqrt{2})^{1/3}}$, thus $MSE^{(o)}(u) = \frac{19}{48(8\sqrt{2})^{2/3}}u^2 \approx 0.0785u^2$.

As a result, $MSE^{(s)} \approx 0.0833u^2$, $MSE^{(h)} \approx 0.0801u^2$ and $MSE^{(o)} \approx 0.0785u^2$, where the volume of the grid is u. We can thus conclude that $\forall u \in \mathbb{R}^+, MSE^{(o)}(u) < MSE^{(h)}(u) < MSE^{(s)}(u)$.

C Proof of Theorem 2

Proof. Let us rewrite the end-to-end loss in equation 1 as an unconstrainted multi-objective optimization:

$$\frac{\mathcal{L}}{2+\lambda} = \mathop{\mathbb{E}}_{\mathbf{x} \sim p_x} \left[-\frac{1}{2+\lambda} log(p_f(\hat{\mathbf{z}}; \Psi)) - \frac{1}{2+\lambda} log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)) + \frac{\lambda}{2+\lambda} d(\mathbf{x}, \hat{\mathbf{x}}) \right].$$

Where $\alpha_1 = 1/(2+\lambda)$, $\alpha_2 = 1/(2+\lambda)$ and $\alpha_3 = \lambda/(2+\lambda)$. Since $\lambda > 0, \forall i, \alpha_i > 0$ and $\sum_i \alpha_i = 1$, the set of α_is corresponds to an unconstrained multi-objective optimization's coefficients set.

Since we do not target to update the parameters of the end-to-end model, but just target to adjust main and side latents $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$, we keep all the parameter fixed, but $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ as variable. Thus, we can represent

side information's bitlength objective with $\mathcal{L}_1(\hat{\mathbf{z}}) := -log(p_f(\hat{\mathbf{z}}; \Psi))$, main information's bitlength objective with $\mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{z}}) := -log(log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)))$ and finally distortion objective with $\mathcal{L}_3(\mathbf{x}, \hat{\mathbf{y}}) := -d(\mathbf{x}, g_s(\hat{\mathbf{y}}; \theta))$. With these notations, training the model can be written as an unconstrained multi objective optimization problem:

$$\hat{\mathbf{y}}^*, \hat{\mathbf{z}}^* = \arg\min_{\hat{\mathbf{y}}, \hat{\mathbf{z}}} \left(\mathbb{E}_{\mathbf{x} \sim p_x} \left[\alpha_1 \mathcal{L}_1(\hat{\mathbf{z}}) + \alpha_2 \mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{z}}) + \alpha_3 \mathcal{L}_3(\mathbf{x}, \hat{\mathbf{y}}) \right] \right)$$

This problem has two set of variables to be optimized, and both variables should meet KKT conditions. The variable $\hat{\mathbf{z}}$ is decorrelated from $\mathcal{L}_3(\mathbf{x}, \hat{\mathbf{y}})$, thus $\nabla_{\hat{\mathbf{z}}} \mathcal{L}_3(\mathbf{x}, \hat{\mathbf{y}}) = 0$. We can write KKT condition for $\hat{\mathbf{z}}$ as follows.

$$\mathbb{E}_{\mathbf{x} \sim p_x} \left[\alpha_1 \nabla_{\hat{\mathbf{z}}} \mathcal{L}_1(\hat{\mathbf{z}}) + \alpha_2 \nabla_{\hat{\mathbf{z}}} \mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{z}}) \right] = 0.$$

Since $\alpha_1 = \alpha_2$, we obtain the first condition in equation 3 by simply replacing $\mathcal{L}_1(\hat{\mathbf{z}})$ and $\mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ with their definitions.

The second variable $\hat{\mathbf{y}}$ does not impact $\mathcal{L}_1(\hat{\mathbf{z}})$, thus $\nabla_{\hat{\mathbf{y}}} \mathcal{L}_1(\hat{\mathbf{z}}) = 0$. We can write KKT condition for $\hat{\mathbf{y}}$ as follows.

$$\mathbb{E}_{\mathbf{x} \sim p_x} \left[\alpha_2 \nabla_{\hat{\mathbf{y}}} \mathcal{L}_2(\hat{\mathbf{y}}, \hat{\mathbf{z}}) + \alpha_3 \nabla_{\hat{\mathbf{y}}} \mathcal{L}_3(\mathbf{x}, \hat{\mathbf{y}}) \right] = 0.$$

If we replace the objectives by their definitions and divide the two-hand side by α_2, and since $\alpha_3/\alpha_2 = \lambda$, we reach the second condition in the theorem in equation 4. □

D Ablation Studies

In this section, we present several ablation studies for uniform VQ and latent shifting.

D.1 Gain Analysis of Latent Shift

Table 3: Upper limits of the gradient based latent shifting on **mbt2018-mean** codec.

	Only Side Shift (BD-Rate)	Only Main Shift (BD-Psnr)	Only Main Shift (BD-Rate)	Main & Side Shift (BD-Rate)
True Gradients	-1.011%	1.3972 dB	-25.139%	-26.150%
Latent Shift	-0.031%	0.0705 dB	-1.270%	-1.301%

To understand the upper limits of gradient based latent shifting, we use the true gradients instead of proxy one and measure the performance. Simply, we shift the side latents by $\hat{\mathbf{z}} \leftarrow \hat{\mathbf{z}} + \rho_f^* \nabla_{\hat{\mathbf{z}}}(-log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)))$ after decoding $\hat{\mathbf{z}}$. Later, we shift the main latent by $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \rho_h^* \nabla_{\hat{\mathbf{y}}}(d(\mathbf{x}, g_s(\hat{\mathbf{y}}; \theta)))$ after decoding $\hat{\mathbf{y}}$. We can see this hypothetical case as if the correlations are -1. This case is mentioned by **True Gradients** in the results on Table 3 while our proposal is **Latent Shift**. The results are taken with **mbt2018-mean** image codec on Kodak dataset.

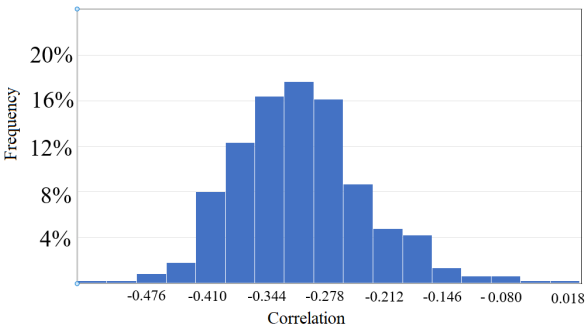


Figure 6: Histogram of correlation between gradients wrt main latents. The data is taken with **mbt2018-mean** image codec on Kodak and Clic dataset.

According to this results, we can see that even if the side latent's gradients were perfectly correlated, our maximum gain would be around 1%. Since the correlation of gradients wrt the side latent is weak ($r^2 \approx -0.07$), our gain is negligible. As a consequence, in practice, shifting side latent may not be neglected. On the other hand, main latents true gradients increase PSNR by 1.40dB in average which is equivalent of saving around 25% of the bitstream. Our proposal could increase the PSNR by 0.07dB in

average which is equivalent to saving 1.27% of the bitstream for the same quality thanks to the existing correlation between gradients wrt main latents as shown in Figure 6. The gain is of course smaller than the upper limit, but significant still. Since keeping these gradients are costly (nearly the same cost of saving image itself), searching more effective way of using those gradient is meaningful.

D.2 Latent Shift versus Alternatives

In order to show how proposed latent shifting is better than alternatives, we shift the latent in random direction at encoding time such that $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \epsilon(\rho_h)$, where $\epsilon(\rho_h) \in \mathbb{R}^{m \times m \times o} \sim \mathcal{N}(0, 1)$ and ρ_h is a random seed number to be signalled. The best random seed and gradients, in terms of PSNR improvement, should be found at encoding time. We generate 1024 random gradient and encode the best random seed with 10 bits as an extra information. Even though this approach is costly in terms of computational complexity in encoding time, (it needs 1024 times forward pass), we think this is a natural baseline to our proposal, and we refer to it as **Random Shift**.

Another alternative would be to use constant gradient for all latents in a given image. Thus, at encoding time we need to test a large set of values and assume all latents should be shifted by this number such that $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \rho_h$ where $\rho_h \in \mathbb{R}$. The best value of ρ_h should be signaled to the decoder with 10 bit extra cost. We refer to this as **Scalar Shift** approach.

Our last alternative does not use gradients directly, but the fact that they may be correlated as prior knowledge. Particularly, this approach shifts the latent in the opposite direction of the distribution's center such that $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} - \rho_h \operatorname{sign}(\hat{\mathbf{y}} - \boldsymbol{\mu})$. In hyperprior entropy models, the latent is assumed to obey a Gaussian distribution. thus, the latent's entropy gets smaller if it moves towards the center of the distribution. By shifting the latent in opposite of this direction, the entropy increases. Since the factorized entropy model does not use Gaussian distribution, we shift the latent to the opposite direction of zero center means assuming $\boldsymbol{\mu} = 0$ in factorized entropy. The best value of ρ_h should be signaled to the decoder with 10 bit extra cost. We refer to this approach as **Sign Shift**.

Table 4: Average BD-PSNR of **Latent Shift** and some alternatives.

Baseline Codec	Random Shift	Scalar Shift	Sign Shift	Latent Shift
bmshj2018-factorized	0.0002 dB	0.0036 dB	0.0151 dB	0.0297 dB
mbt2018-mean	0.0006 dB	0.0007 dB	0.0339 dB	0.0705 dB

In Table 4, we compare **bmshj2018-factorized** (lowest correlation between gradients) and **mbt2018-mean** (highest correlation between gradients). Since all alternatives need extra 10 bits signaling cost, we neglect it assume the bitlengths are the same as the baselines, and we report the BD-PSNR defined in Bjontegaard (2001). These results show that our proposal is significantly better than all alternatives. The closest one (**Sign Shift**) reaches half of **Latent Shift**'s performance. The random alternatives could not improve the baseline significantly, even though their encoder is almost 1000 times computational demanding.

D.3 Truncated Octahedron versus Hexagon

Concerning uniform VQ quantizations, either hexagonal grid quantization **Hex-Quant** or truncated octahedron grid quantization **Hex-Quant** give different results. Better quantization leads to the smaller quantization error on latent and better reconstruction. However, bigger symbol dictionary is needed (square of SQ's dictionary size in hexagonal grid). Since the arithmetic encoder has a fixed bit resolution, it has representation limitations (for 16-bit resolution, the minimum probability is 1/65536). Thus, in practice, we can assign 1/65536 probability to the symbol whose probability is lower than 1/65536, what makes encoding less efficient and increase the rate. Another alternative is to remove those symbols, what increases the quantization error of latents but decreases the rate. In practice, we have chosen to remove the symbol if its probability is less than 10^{-7}, and we found out that it always gives better RD performance.

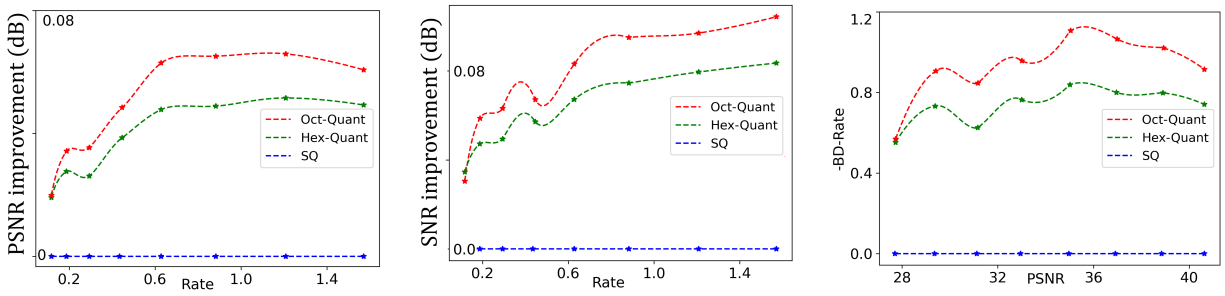
In this ablation study, we analyse further the quantization effects on reconstruction error (PSNR), as well as the impact of latent quantization error in terms of Signal Noise Ration (SNR) w.r.t. the rate (under the

assumption that arithmetic encoder has infinite bit resolution). To this end, we do not encode the symbols into bitstream, but calculate the lower bound of bitlength according to Shanon's entropy theorem without limits of integer resolution that dictates some certain probabilities on symbols. We calculate SNR of latents by $SNR = -10log_{10}(d(\hat{\mathbf{y}}, \mathbf{y}))$ where $\mathbf{y}$ is latent, $\hat{\mathbf{y}}$ is reconstructed latent by certain quantization techniques and $d(.,.)$ measures MSE error between two inputs.

We use **mbt2018-mean** neural codec in Minnen et al. (2018) on Kodak dataset. Results are presented in Table 5 and show that **Oct-Quant** gives lower reconstruction error (higher PSNR improvement) and lower quantization error over latent (higher SNR improvement) than **Hex-Quant** for the same rate. In addition, **Oct-Quant** saves more bitlength than **Hex-Quant** for the same reconstruction quality. Figure 7 compares the performances of **Oct-Quant** and **Hex-Quant** to the uniform SQ baseline, for different rate and reconstruction quality.

Table 5: Performance of **Hex-Quant** and **Oct-Quant** compare to uniform SQ.

Quantization	BD-PSNR	BD-SNR	BD-Rate
Hex-Quant	0.0374 dB	0.0600 dB	-0.748%
Oct-Quant	0.0480 dB	0.0736 dB	-0.957%



(a) (b) (c)

Figure 7: BD-Psnr, BD-Snr and -Bd-Rate performance of Hex-Quant and Oct-Quant compare to the uniform SQ. In all perspective, Oct-Quant is the best, while Hex-Quant comes for the second.

E Numeric PMF Calculation for Hexagonal Domain

In this section, we give some implementation details.

The closed form solution for the integrals of multidimensional known densities over any domain has very specific solutions, and generally no tractable form (Savaux & Le Magoarou, 2020). That holds even for Gaussian distribution of dimension 2 on the regular hexagon domain. Thus, there is no closed form solution of equation 2, where the probabilities are independent Gaussian on the domain of regular hexagonal. However, we find the solution using the combination of both analytic and numerical integration as detailed below.

The integral centred around (x, y) of hexagonal domain G for the independent Gaussian distribution is given by

$$P(x, y) = \int_G N(x, \mu_1, \sigma_1) * N(y, \mu_2, \sigma_2) dxdy. \tag{10}$$

For sake of simplicity, let $(x, y) = (0, 0)$ and a be the one side length of the hexagonal. Then, the integral is defined as

$$P(x,y) = \int_{0}^{a\frac{\sqrt{3}}{2}} \int_{-a+\frac{y}{\sqrt{3}}}^{a-\frac{y}{\sqrt{3}}} N(x,\mu_1,\sigma_1) * N(y,\mu_2,\sigma_2)dxdy + \int_{-a\frac{\sqrt{3}}{2}}^{0} \int_{-a-\frac{y}{\sqrt{3}}}^{a+\frac{y}{\sqrt{3}}} N(x,\mu_1,\sigma_1) * N(y,\mu_2,\sigma_2)dxdy, \quad (11)$$

where the hexagon is divided into lower and upper half parts, and we sum the two integrals. It is noted that the integral does not admit a closed form solution, but the inner integral has the closed form and outer integral does not have the analytic solution. The solution of the inner integral in the first part is given by

$$\int_{-a+\frac{y}{\sqrt{3}}}^{a-\frac{y}{\sqrt{3}}} N(x,\mu_1,\sigma_1)dx = \frac{1}{2} erf\left(\sqrt{2}\frac{(-\mu_1 + x)}{2 * \sigma_1}\right)\Big|_{x=-a-\frac{y}{\sqrt{3}}}^{x=a-\frac{y}{\sqrt{3}}} \quad (12)$$

by substituting into equation 11, we obtain:

$$\int_{0}^{a\frac{\sqrt{3}}{2}} \int_{-a+\frac{y}{\sqrt{3}}}^{a-\frac{y}{\sqrt{3}}} N(x,\mu_1,\sigma_1) * N(y,\mu_2,\sigma_2)dxdy = \int_{0}^{a\frac{\sqrt{3}}{2}} \frac{1}{2} erf\left(\sqrt{2}\frac{(-\mu_1 + x)}{2 * \sigma_1}\right)\Big|_{x=-a-\frac{y}{\sqrt{3}}}^{x=a-\frac{y}{\sqrt{3}}} * N(y,\mu_2,\sigma_2)dy \quad (13)$$

This is finally solved by the numerical integration[4]. Similarly, we can also obtain the integral of the second part.

F Complexity Analysis

Here we describe the computational complexity of proposed method and provide detailed analysis of source of the additional computational costs and the run time results of proposals with respect to the baseline models.

F.1 Source of complexity

The complexity of our proposed method includes encoding and decoding complexity of uniform VQ, and encoding and decoding complexity of Latent shift, and they are detailed below.

F.1.1 Encoding complexity of uniform VQ

First, we should define our grid centers as codebook such that $\mathbf{c}^{(i)} \in R^v, i = 1 \dots M$ be the M grid centers of the v-dimensional shape (e.g hexagonal grids center for $v = 2$) and PMF of our v-dimensional grid using learned $1D$ PMF by equation 2 in the main paper.

The additional steps involved in the encoding time over the baseline approach are as follows:

1. Reshape the latents into pseudo v-dimensional vector $\mathbf{y} \in R^{m \times m \times o} \rightarrow \check{\mathbf{y}} \in R^{b \times v}$, where $b = \frac{m.n.o}{v}$ (before reshaping, the latent can be sorted by their distribution's σ parameters. In this way, the latents that belongs to a similar distribution can be dropped into the same latent vector).
2. Find closest codebook from our initial quantization grids for each b vector such $\tilde{\mathbf{y}}_j = argmin_i ||\check{\mathbf{y}}_j - \mathbf{c}^{(i)}||$, where $\tilde{\mathbf{y}}_j \in \{1 \dots M\}, j = 1 \dots b$ are the codes to be encoded into bitstream.

In the shared step (with baseline method), $\tilde{\mathbf{y}}_j, j = 1 \dots b$ should be encoded into bitstream by given PMF. Thus, in encoding time above 2 step's complexity is the reason for the extra complexity introduced by our proposed method.

[4] https://github.com/esa/torchquad

F.1.2 Decoding complexity of uniform VQ

In decoding time, $\tilde{\mathbf{y}}_j, j = 1 \dots b$ should be decoded from bitstream by provided PMF table. The source of extra complexity of our method (over the baseline) in the decoding time is the following two steps:

1. De-quantizate the decoded codes: find the center of selected quanta center such that $\bar{\mathbf{y}}_j = \mathbf{c}^{\tilde{\mathbf{y}}_j}, j = 1 \dots b$
2. Reshape dequantized latent into the original dimensions such that $\bar{\mathbf{y}} \in R^{b \times v} \rightarrow \hat{\mathbf{y}} \in R^{m \times m \times o}$, (if the ordering wrt σ is applied, we need to revert the order back in order to have the latents in original order)

In shared step (with the baseline method), these dequantized and reshaped latents are fed to the decoder network.

F.1.3 Encoding complexity of Latent Shift

In encoding time, the complexity introduced by the *Latent Shift* are as follows:

1. The calculation of the gradients of the entropy $\nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)); \theta)$
2. finding the best step size ρ_h that maximize the reconstruction quality when the latent is shifted as $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + \rho_h^* \nabla_{\hat{\mathbf{y}}}(-log(p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta)))$

It is noted that calculating gradient of the entropy has a negligible complexity, because we do not need the forward pass and there is closed form solution. When the entropy model uses Gaussian distribution such as $p_h(\hat{\mathbf{y}}; \hat{\mathbf{z}}, \Theta) := N(\hat{\mathbf{y}}; \mu, \sigma)$, the gradient of the entropy becomes the derivative of $-log(N(\hat{\mathbf{y}}; \mu, \sigma))$ wrt $\hat{\mathbf{y}}$ which has closed form solution, where $N(.; \mu, \sigma)$ is PDF of gaussian distribution by given fixed μ, σ parameters.

However, the second step to find the best step size ρ_h is moderately demanding process. In the experiments, we tested 8 different choices for p_h and selected the best one. Thus, it needs 8 forward passes of decoding and calculation of error between input and reconstruction.

F.1.4 Decoding complexity of latent shift

In decoding time, the complexity of decoding the latents from the bitstream is already included in the decoding step of uniform VQ. So, the decoding complexity of the *Latent Shift* are: (1) decoding a single scalar ρ_h from bitstream and calculating the gradient entropy wrt latents, which are both negligible; (2) Shifting the latent by using gradients and decoded step size which is actually adding two tensor, and it is negligible compared to the decoding complexity of the baseline model.

F.2 Runtime complexity analysis

For experimental computational complexity, we run the test on single core of Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz with preventing multi-thread operations. We measured the computational time of *cheng2020_attn*, *mbt2018-mean* on Kodak dataset and SSF on bunny dataset. In order to have the same resolution image, we just crop the top-left part of the frames in bunny frame set which has the same resolution as kodak dataset (512x768). In this way, all methods complexity can be comparable between each other.

The results show average encoding/decoding time in seconds for the provided lowest bitrate of baseline model and our Hex-Quant, Oct-Quant and Latent Shift. The reported encoder running times is the duration from the input image is read to the memory to the producing the bitstream including arithmetic encoder time. It is the same for decoding. The duration is from the bitstream in the memory to obtain reconstructed image including arithmetic decoder process. The results are reported in the table 6.

Model	Encoding (in secs)				Decoding (in secs)			
	Base	Hex Quant	Oct Quant	Latent Shift	Base	Hex Quant	Oct Quant	Latent Shift
mbt2018-mean	0.5274s	0.5436s (+3%)	0.5548s (+5%)	5.5674s (x10.1)	0.7138s	0.7264s (+1.8%)	0.7265s) (+1.8%)	0.7188s (+0.7%)
cheng2020-attn	3.4982s	3.5238s (+0.7%)	3.5368s (+1.1%)	18.5345s (x5.3)	8.3065s	8.3190s (+0.1%)	8.3195s (+0.1%)	8.3115s (+0.06%)
SSF	1.8618s	1.9068s (+2.4%)	1.9230s (+3.2%)	11.1418s (x6.0)	1.3025s	1.3285s (+1.9%)	1.3288s (+1.9%)	1.3125s (+0.07%)

Table 6: Encoding and decoding running time (in seconds) of the baseline method and our proposed method (either Hex Quant or Oct quant + latent shift) for the lowest bit-rate model of image and video codec. The numbers in the bracket indicates the additional complexity introduced by our method over the baseline.

The main source of additional encoding complexity of Hex-Quant and Oct-Quant is to find the closest codebook. The naive search gives around %20 overhead complexity for 3-dimensional VQ. But there are algorithms that find nearest quanta center very efficiently as defined in (Agrell et al., 2002; Conway & Sloane, 1982). We have adapted this solution to our model and decreased additional cost to 3%-5% in encoding time for *mbt2018-mean* model. Though *cheng2020_attn* model is the most complex model in our studied neural codec, our extra computation cost is almost the same with *mbt2018-mean* model (our Hex-Quant and Oct-Quant runtime depends on the resolution of the latent not the architecture), Hex-Quant and Oct-Quant's relative overhead is between 1.1%-0.7% in encoding time.

For SSF, we neglect the I frame compression runtime (because it is the same with *mbt2018-mean*) but measured the average runtime for encoding and decoding of P frames which is done by two VAEs, one for motion information another for residual information. Thus, there are two main information (main motion and main residual) and our quantization should be done on these latent. It explains our extra absolute runtime is more than absolute extra runtime in *mbt2018-mean*. Since the P frame compression model is more complex than *mbt2018-mean*, Hex-Quant and Oct-Quant's extra relative runtime is between 2.4%-3.2%. Since the differences between dimension of VQ is just reshaping tensor and number of indices to be found from dictionary in decoding time, the decoding time extra complexity of Hex-Quant and Oct-Quant is negligible and almost the same with hex and oct quantization as it can be seen in the table 6

Regarding the *Latent Shift*, its decoding complexity is less than 1% as it can be seen in Table 6. However, *Latent Shift*'s encoding complexity is between 5-10 magnitude of baseline encoding running time. Almost all complexity comes from finding the best shift step size by brutal force out of 8 candidates.

G Additional Results

In this section, we present additional results of gain evaluated over different methods with Kodak and Clic-2021 dataset. Figure 8-11 shows th BD-rate gain for **bmshj2018-factorized** in Ballé et al. (2017), **mbt2018-mean** and **mbt2018** in Minnen et al. (2018) and **cheng2020-attn** in Cheng et al. (2020).

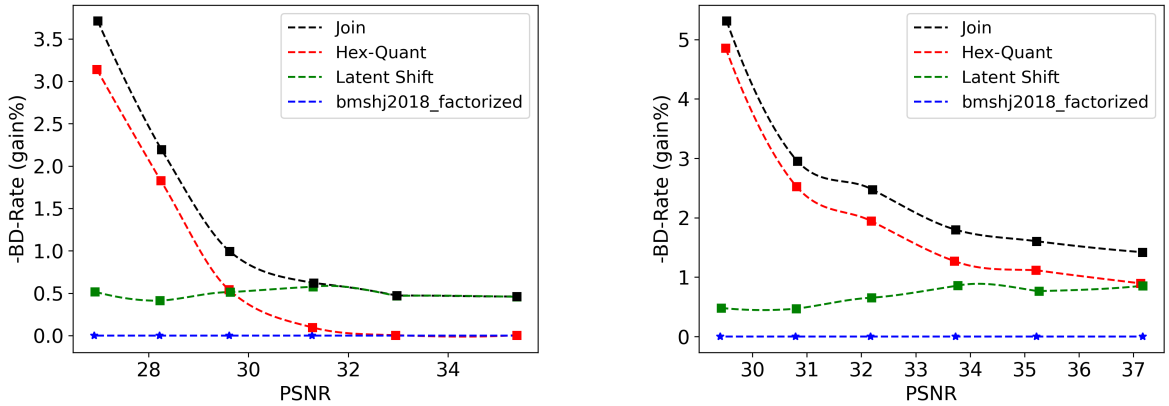


Figure 8: BD-Rates gain of our proposals from **bmshj2018-factorized** codecs for different quality a) Kodak test set b) Clic-2021

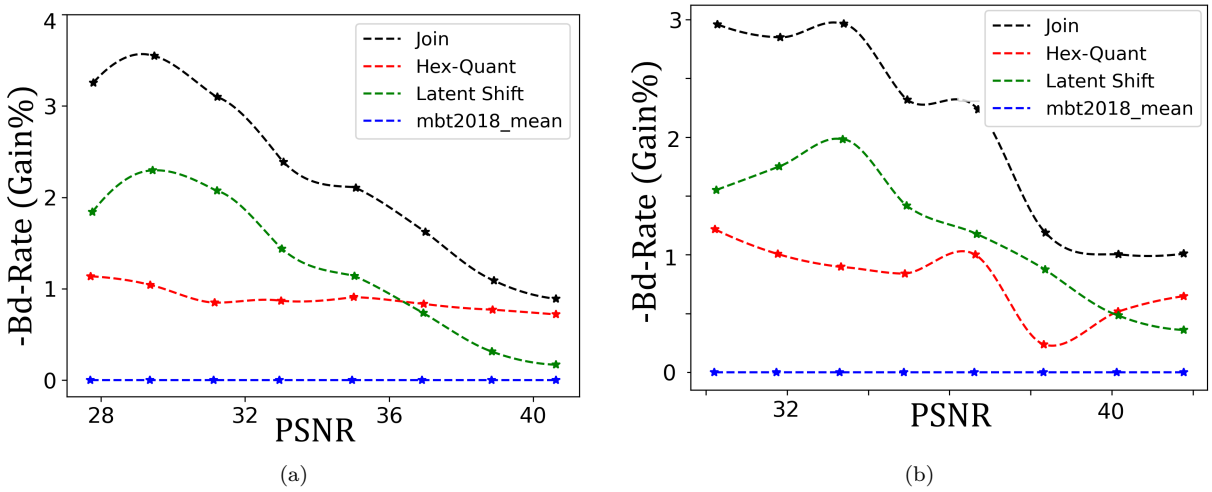


Figure 9: BD-Rates gain of our proposals from **mbt2018-mean** codecs for different quality a) Kodak test set b) Clic-2021

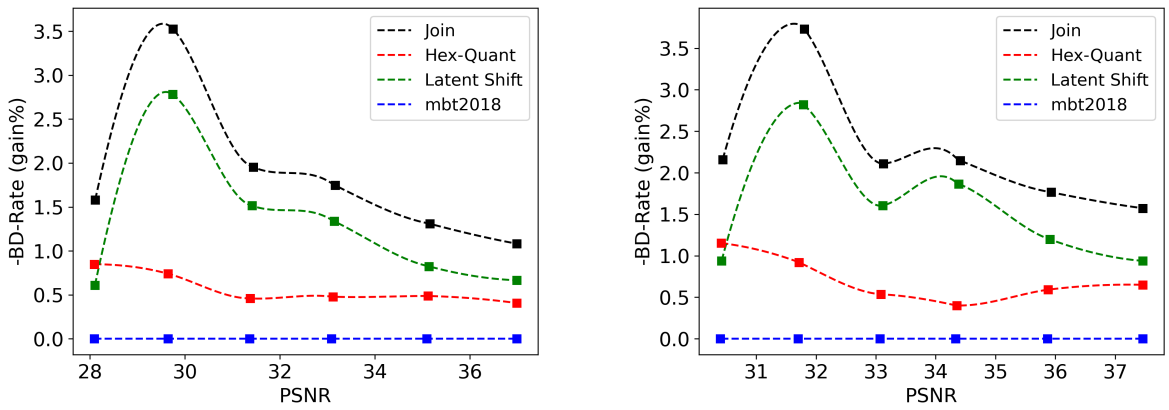


Figure 10: BD-Rates gain of our proposals from **mbt2018** codecs for different quality a) Kodak test set b) Clic-2021

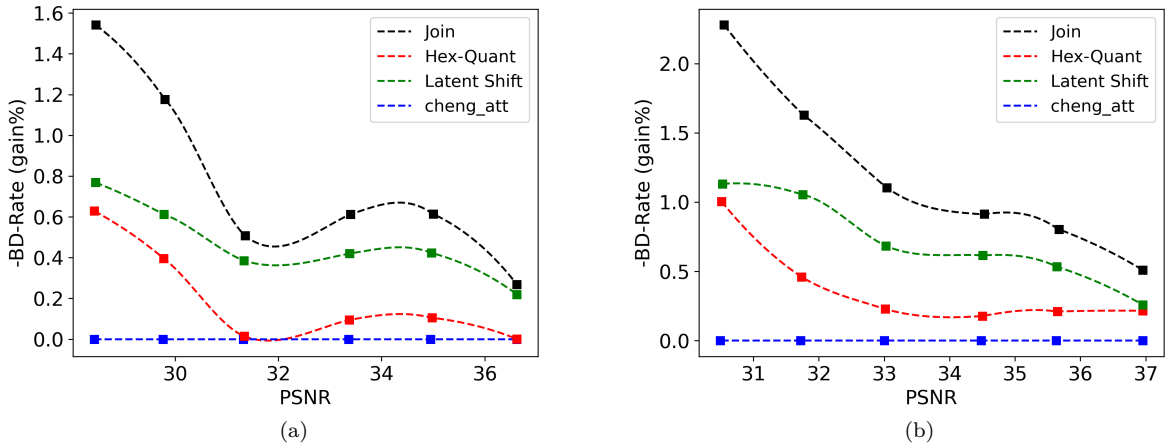


Figure 11: BD-Rates gain of our proposals from **cheng2020-attn** codecs for different quality a) Kodak test set b) Clic-2021