Bit Allocation using Optimization

Tongda Xu 1 Han Gao 2,3 Chenjian Gao 2,4 Yuanyuan Wang 2 Dailan He 2 Jinyong Pi 2 Jixiang Luo 2 Ziyu Zhu 5 Mao Ye 3 Hongwei Qin 2,6 Yan Wang 1 Jingjing Liu 1,7 Ya-Qin Zhang 1,5,7

Abstract
In this paper, we consider the problem of bit allocation in Neural Video Compression (NVC). First, we reveal a fundamental relationship between bit allocation in NVC and Semi-Amortized Variational Inference (SAVI). Specifically, we show that SAVI with GoP (Group-of-Picture)-level likelihood is equivalent to pixel-level bit allocation with precise rate & quality dependency model. Based on this equivalence, we establish a new paradigm of bit allocation using SAVI. Different from previous bit allocation methods, our approach requires no empirical model and is thus optimal. Moreover, as the original SAVI using gradient ascent only applies to single-level latent, we extend the SAVI to multi-level such as NVC by recursively applying back-propagating through gradient ascent. Finally, we propose a tractable approximation for practical implementation. Our method can be applied to scenarios where performance outweights encoding speed, and serves as an empirical bound on the R-D performance of bit allocation. Experimental results show that current state-of-the-art bit allocation algorithms still have a room of $\approx 0.5$ dB PSNR to improve compared with ours. Code is available at https://github.com/tongdaxu/Bit-Allocation-Using-Optimization.

1. Introduction
Neural Video Compression (NVC) has been an active research area. Recently, state-of-the-art (SOTA) NVC approaches (Hu et al., 2022; Li et al., 2022a) have achieved comparable performance with advanced traditional video coding standards such as H.266 (Bross et al., 2021). The majority of works in NVC focus on improving motion representation (Lu et al., 2019; 2020b; Agustsson et al., 2020) and better temporal context (Djelouah et al., 2019; Lin et al., 2020; Yang et al., 2020a; Yilmaz & Tekalp, 2021; Li et al., 2021). However, the bit allocation of NVC is relatively under-explored (Li et al., 2022b).

The target of video codec is to minimize R-D (Rate-Distortion) cost $R + \lambda D$, where $R$ is bitrate, $D$ is distortion and $\lambda$ is the Lagrangian multiplier controlling R-D trade-off. Due to the frame reference structure of video coding, using the same $\lambda$ for all frames/regions is suboptimal. Bit allocation is the task of solving $\lambda$ for different frames/regions. For traditional codecs, the accurate bit allocation has been considered intractable. And people solve $\lambda$ approximately via empirical rate & quality dependency model (Li et al., 2014; 2016) (See details in Sec. 2.2).

The pioneer of bit allocation for NVC (Rippel et al., 2019; Li et al., 2022b) adopts the empirical rate dependency from Li et al. (2014) and proposes a quality dependency model based on the frame reference relationship. More recently, Li et al. (2022a) propose a feed-forward bit allocation approach with empirical dependency modeled implicitly by neural network. However, the performance of those approaches heavily depends on the accuracy of empirical model. On the other hand, we show that an earlier work, Online Encoder Update (OEU) (Lu et al., 2020a), is in fact also a frame-level bit allocation for NVC (See Appendix. E). Other works adopt simplistic heuristics such as fixed $\lambda$ schedule to achieve very coarse bit allocation (Cetin et al., 2022; Hu et al., 2022; Li et al., 2023).

In this paper, we first examine the relationship of bit allocation in NVC and Semi-Amortized Variational Inference (SAVI) (Kim et al., 2018; Marino et al., 2018). We prove that SAVI using GoP-level likelihood is equivalent to pixel-level bit allocation using precise rate & quality dependency model. Based on this relationship, we propose a new paradigm of bit allocation using SAVI. Different from previous bit allocation methods, this approach achieves pixel-level control and requires no empirical model. And thus, it is optimal assum-
ing gradient ascent can achieve global maxima. Moreover, as the original SAVI using gradient ascent only applies to single-level latent variable, we extend SAVI to latent with dependency by recursively applying back-propagating through gradient ascent. Furthermore, we provide a tractable approximation to this algorithm for practical implementation. Despite our approach increases encoding complexity, it does not affect decoding complexity. Therefore, it is applicable to scenarios where R-D performance is more important than encoding time. And it also serves as an empirical bound on the R-D performance of other bit allocation methods. Experimental results show that current bit allocation algorithms still have a room of \( \approx 0.5 \) dB PSNR to improve, compared with our results. Our bit allocation method is compatible with any NVC method with a differentiable decoder. And it can be even directly adopted on existing pre-trained models.

To wrap up, our contributions are as follows:

- We prove the equivalence of SAVI on NVC with GoP-level likelihood and pixel-level bit allocation using the precise rate & quality dependency model.
- We establish a new paradigm of bit allocation algorithm using gradient based optimization. Unlike previous methods, it requires no empirical model and is thus optimal.
- We extend the original SAVI to latent with general dependency by recursively applying back-propagating through gradient ascent. And we further provide a tractable approximation so it scales to practical problems such as NVC.
- Empirical results verify that the current bit allocation algorithms still have a room of \( \approx 0.5 \) dB PSNR to improve, compared with our optimal results.

2. Preliminaries

2.1. Neural Video Compression

The majority of NVC follow a mixture of latent variable model and temporal autoregressive model (Yang et al., 2020a). To encode \( i^{th} \) frame \( x_i \in \mathbb{R}^{HW} \) with \( HW \) pixels inside a GoP \( x_{1:N} \) with \( N \) frames, we first transform the \( i^{th} \) frame in context of previous frames to obtain latent parameter \( y_i = f_\phi(x_i, |y_{<i}|) \), where \( f_\phi(\cdot) \) is the encoder (inference model) parameterized by \( \phi \). \( \cdot \) is the rounding operator, and \( |y_{<i}| \) is the quantized latent of previous frames. Then, we obtain \( |y_i| \) by quantizing the latent \( y_i \). Next, an entropy model \( p_\theta(|y_i| | |y_{<i}|) \) parameterized by \( \theta \) is used to evaluate the probability mass function (pmf, prior) \( P_\theta(|y_i| | |y_{<i}|) = F_\theta(|y_i| + 0.5| |y_{<i}|) - F_\theta(|y_i| - 0.5| |y_{<i}|) \), where \( F_\theta \) is the cdf of \( p_\theta \). With the pmf, we encode \( |y_i| \) with bitrate \( R_i = -\log P_\theta(|y_i| | |y_{<i}|) \).

During decoding process, we obtain the reconstruction \( \hat{x}_i = g_\theta(|y_{<i}|) \), where \( g_\theta(\cdot) \) is the decoder (generative model) parameterized by \( \theta \). Finally, we compute distortion \( D_i = d(x_i, \hat{x}_i) \), which can be interpreted as the data likelihood \( \log P_\theta(x_i | |y_{<i}|) \) so long as we treat \( D_i \) as energy of Gibbs distribution (Minnen et al., 2018). For example, when \( d(\cdot, \cdot) \) is mean square error (MSE), we can interpret \( d(x, \hat{x}) = -\log p_\theta |x| |y_i\rangle + C \), where \( p_\theta |x| |y_i\rangle = \mathcal{N}(\hat{x}, 1/2\lambda, I) \). The optimization target is the frame-level R-D cost (Eq. 3) with Lagrangian multiplier \( \lambda \) controlling R-D trade-off. In this paper, we consider pixel-level distortion and Lagrangian multiplier, and we denote \( D_i \in \mathbb{R}^{HW} \) as the pixel-level Lagrangian multiplier, and \( \lambda^2 D_i \) as the weighted distortion. The above procedure can be described by:

\[
\begin{align}
\hat{y}_i &= f_\phi(x_i, \tilde{y}_{<i}), \quad \text{where} \quad \tilde{y}_i = |y_i|, \\
R_i &= -\log P_\theta(\tilde{y}_i | \hat{y}_{<i}), \\
\phi^*, \theta^* &\leftarrow \arg\min_{\phi, \theta} R_i + \lambda^2 D_i. 
\end{align}
\]

As the rounding operation \( \lfloor \cdot \rfloor \) is not differentiable, Ballé et al. (2016) propose to relax it by additive uniform noise (AUN), and replace \( |y_i| \) with \( \tilde{y}_i = y_i + U(-1/2, 1/2) \) during training. Under such formulation, we can also treat the above encoding-decoding process as a Variational Autocoder (Kingma & Welling, 2013) on graphic model \( \tilde{y}_{<i} \to x_i \) with variational posterior:

\[
\begin{align}
q_\phi(\tilde{y}_{1:N} | x_{1:N}) &= \prod_{i=1}^N q_\phi(\tilde{y}_i | x_i, \tilde{y}_{<i}), \\
q_\phi(\tilde{y}_i | x_i, \tilde{y}_{<i}) &= U(y_i - 1/2, y_i + 1/2). 
\end{align}
\]

And then minimizing the uniform noise relaxed R-D cost (Eq. 3) is equivalent to maximizing the evident lowerbound (ELBO) (Eq. 5) by Stochastic Gradient Variational Bayes-A (SGVB-A) (Kingma & Welling, 2013):

\[
\mathcal{L}_i = \mathbb{E}_{q_\phi(\tilde{y}_i | x_i, \tilde{y}_{<i})} [\log P_\theta(\tilde{y}_i | \tilde{y}_{<i}) + \log p_\theta(x_i | \tilde{y}_{<i})] - R_i - \lambda D_i.
\]

2.2. \( \lambda \)-Domain Bit Allocation

In this section, we briefly review the main results of \( \lambda \)-domain bit allocation (Li et al., 2016) (See detailed derivation in Appendix. A). Consider the task of compressing \( x_{1:N} \) with a overall R-D trade-off parameter \( \lambda_0 \). Then the target of pixel-level bit allocation is to minimize GoP-level R-D cost by adjusting the pixel-level R-D trade-off parame-
As we can solve the pixel-level R-D trade-off parameter as:

$$\lambda^*_1, \ldots, \lambda^*_N \leftarrow \arg\min_{\lambda_1, \ldots, \lambda_N} \sum_{i=1}^{N} R^*_i + \lambda^*_0 \sum_{i=1}^{N} D^*_i,$$

where $R^*_i, D^*_i \leftarrow \min R_i + \lambda^*_T D_i,$ (6)

where $\lambda_0 = \lambda_0 I$ and $I$ is identity matrix. And following [Li et al., 2016] and [Tishby et al., 2000], we have the optimal conditions for $\lambda_i$ as:

$$\sum_{j=i}^{N} \frac{dR_j}{d\lambda_i} + \lambda^*_0 \sum_{j=i}^{N} \frac{dD_j}{d\lambda_i} = 0,$$

$$\lambda^*_i + \frac{dR_i}{dD_i} = 0.$$ (7)

The conventional $\lambda$-domain bit allocation (Li et al., 2016) introduces empirical models to solve $\lambda_i$. Specifically, it approximates the rate & quality dependency as:

$$\frac{dR_j}{dR_i} \approx 0, \sum_{j=i}^{N} \frac{dD_j}{d\lambda_i} \approx \omega_i I,$$ (9)

where $\omega_i$ is the model parameter. By taking Eq. 9 into Eq. 7: (See details in Appendix. A), we have:

$$\frac{dR_i}{d\lambda_i} + \omega_i \lambda^*_0 \frac{dD_i}{d\lambda_i} \approx 0.$$ (10)

By taking Eq. 8 into Eq. 10: (See details in Appendix. A) we can solve the pixel-level R-D trade-off parameter as:

$$\lambda_i = \omega_i \lambda^*_0 \approx \lambda^*_i.$$ (11)

As $\lambda_i$ is only an approximation to the optimal solution $\lambda^*_i$ in Eq. 6 the performance of $\lambda$-domain bit allocation heavily relies on the correctness of the empirical rate & quality dependency model in Eq. 9.

3. Bit Allocation using Optimization

3.1. Bit Allocation and SAVI

Consider applying SAVI (Kim et al., 2018) to NVC with GoP-level likelihood as target. This means that we initialize the variational posterior parameters $y_{1:N}$ from amortized encoder $f_o(y_{1:N}|x)$, and optimize them directly to maximize GoP-level ELBO or minus GoP-level R-D cost with $\lambda_0$ as trade-off parameter:

$$y_{1:N} \leftarrow \arg\max_{y_{1:N}} \mathcal{L},$$

where $\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}_i = \sum_{i=1}^{N} -(R_i + \lambda_0^T D_i).$ (12)

It is not surprising that this optimization can improve the R-D performance of NVC, as previous works in density estimation (Kim et al., 2018; Marino et al., 2015) and image compression (Yang et al., 2020; Gao et al., 2022) have shown that this approach can reduce the amortization gap of VAE (Cremer et al., 2018), and thus reduce R-D cost.

However, it is interesting to understand its relationship to bit allocation. First, let’s construct a pixel-level bit allocation $\lambda^*_i$ that satisfies:

$$\frac{d}{dy_i} \left( = \frac{\lambda^*_i}{\lambda} \right) \approx \frac{\lambda}{\lambda_i}$$ (13)

We call $\lambda^*_i$ the equivalent bit allocation map, as minimizing the single frame R-D cost $R_i + \lambda^*_i D_i$ is equivalent to maximizing the GoP-level likelihood $\mathcal{L}$. In other words, $\lambda^*_i$ is the bit allocation that is equivalent to SAVI using GoP-level likelihood. Moreover, $\lambda^*_i$ can be solved explicitly, and it is the solution to optimal bit allocation problem of Eq. 6. Formally, we have the following results:

**Theorem 3.1.** The SAVI using GoP-level likelihood is equivalent to bit allocation with:

$$\lambda_i^* = (I + \sum_{j=i+1}^{N} \sum_{j=i+1}^{N} \frac{dD_j}{dD_i} \lambda_0 \sum_{j=i+1}^{N} \frac{dR_j}{dD_i}.$$ (14)

where $dD_j/dD_i, dR_j/dD_i$ can be computed numerically through the gradient of NVC model. (See proof in Appendix. B)

**Theorem 3.2.** The equivalent bit allocation map $\lambda^*_i$ is the solution to the optimal bit allocation problem in Eq. 6. In other words, we have:

$$\lambda^*_i = \lambda_i^*.$$ (15)

(See proof in Appendix. B)

Theorem 3.2 shows that SAVI with GoP-level likelihood is equivalent to pixel-level accurate bit allocation. Compared with previous bit allocation methods, this SAVI-based bit allocation does not require empirical model, and thus its solution $\lambda^*_i$ is accurate instead of approximated. Furthermore, it generalizes the $\lambda$-domain approach’s solution $\lambda_i$ in Eq. 11 (See details in Appendix. A).

Given external $\lambda^*_i$, we can achieve bit allocation by optimizing the frame-level R-D cost. However, it is important to note that $\lambda^*_i$ is intractable, as it requires matrix multiplication between quality dependency model $dD_j/dD_i$). A single precision quality dependency model of a $1280 \times 720$ video costs $3,397 \text{GB}$ to store.

On the other hand, the result in Theorem 3.1 is also correct for traditional codec. However, as $\lambda^*_i$ is intractable and the traditional codec is not differentiable, we can not use SAVI to achieve bit allocation for traditional codec.
3.2. A Naïve Implementation

The most intuitive way to achieve the SAVI/bit allocation described above is directly updating all latent with gradient ascent:

\[ y_i^0 \leftarrow f_\phi(x, y_{<i}^0), \quad y_i^{k+1} \leftarrow y_i^k + \alpha \frac{dL(y_{1:N}^k)}{dy_i^k}, \]

where \( dL(y_{1:N}^k)/dy_i^k = \sum_j \partial L_j(y_{1:N}^k)/\partial y_i^k. \) (16)

More specifically, we first initialize the variational posterior parameter \( y_0^0 \) by fully amortized variational inference (FAVI) encoder \( f_\phi(x, y_{<i}^0). \) Then we iteratively update the \( k^{th} \) step latent \( y_i^k \) with gradient ascent and learning rate \( \alpha \) for \( K \) steps. During this process, all the posterior parameter \( y_{1:N}^k \) are synchronously updated. This procedure is described in Alg. 1.

Algorithm 1 Original SAVI on DAG

\begin{algorithm}
\begin{algorithmic}[1]
\Procedure{solve-original(x)}{initialize \( y_1^0, \ldots, y_N^0 \leftarrow f_\phi(x) \) from FAVI.}
\For{\( k = 0 \) to \( K - 1 \)}
\For{\( i = 1 \) to \( N \)}
\State \( y_i^{k+1} \leftarrow y_i^k + \alpha \partial L(y_{1:N}^k)/\partial y_i^k \)
\EndFor
\EndFor
\Return \( y_1^K, \ldots, y_N^K \)
\EndProcedure
\end{algorithmic}
\end{algorithm}

In fact, Alg. 1 is exactly the procedure of original SAVI (Kim et al., 2018) without amortized parameter update. This algorithm is designed for single-level variational posterior (the factorization used in mean-field (Blei et al., 2017)), which means \( q_\phi(y_{1:N}|x; 1:N) = \prod_{i=1}^{N} q_\phi(y_i|x; 1:N). \) However, the variational posterior of NVC has an autoregressive form (See Eq. 4), which means that later frame’s posterior parameter \( y_{>i} \) depends on current frame’s parameter \( y_i \).

3.3. The Problem with the Naïve Implementation

For latent with dependency such as NVC, Alg. 1 becomes problematic. The intuition is, when computing the gradient for current frame’s posterior parameter \( y_i \), we need to consider \( y_i \)’s impact on the later frame \( y_{>i} \). And abusing SAVI on non-factorized latent causes gradient error in two aspects: (1). The total derivative \( dL/\partial y_i \) is incomplete. (2). The total derivative \( dL/\partial y_i \) and partial derivative \( \partial L_j/\partial y_i \) are evaluated at wrong value.

Incomplete Total Derivative Evaluation According to the latent’s autoregressive dependency in Eq. 4 and target \( L \) in Eq. 12, we draw the computational graph to describe the latent dependency as Fig. 1(a) and expand the total derivative

\[ dL/dy_i: \]

\[ \frac{dL(y_{1:N})}{dy_i} = \sum_{j=1}^{N} \left( \sum_{l=i+1}^{j} \frac{\partial L_j(y_{1:j})}{\partial y_i} \frac{\partial L_j(y_{1:j})}{\partial y_i} + \frac{\partial L_j(y_{1:j})}{\partial y_i} \right), \]

(17)

As shown in Eq. 16 and Alg. 1, the naïve implementation treats the total derivative \( dL/\partial y_i \) as the sum of the frame level partial derivative \( \partial L_j/\partial y_i \), which is the direct contribution of frame \( j^{th} \) latent \( y_j \) to \( j^{th} \) frame’s R-D cost \( L_j \) as marked in Eq. 17. This incomplete evaluation of gradient signal brings sub-optimality.

Incorrect Value to Evaluate Gradient Besides the incomplete gradient issue, Alg. 1 simultaneously updates all the posterior parameter \( y_{1:N} \) with gradient evaluated at the same step \( y_{1:N}^k \). However, to evaluate the gradient of \( y_i \), all its descendant latent \( y_{>i} \) should already complete all \( K \) steps of gradient ascent. Moreover, once \( y_i \) is updated, all its descendant latent \( y_{>i} \) should be re-initialized by FAVI. Specifically, the correct value to evaluate the gradient is:

\[ y_i^{k+1} \leftarrow y_i^k + \alpha \frac{dL(y_{1:N}^k)}{dy_i^k}, \]

where \( y_i^0 = f_\phi(x, y_{1:N}^0) \), (18)

where \( y_i^k \) denotes the latent \( y_i \) after \( k \) steps of update. In next section, we show how to correct both of the above-mentioned issues by recursively applying back-propagating through gradient ascent (Domke, 2012).

3.4. An Accurate Implementation

Accurate SAVI on 2-level non-factorized latent We first extend the original SAVI on 1-level latent (Kim et al., 2018) to 2-level non-factorized latent. As the notation in NVC, we denote \( x \) as evidence, \( y_1 \) as the variational posterior parameter of the first level latent \( \tilde{y}_1, y_2 \) as the variational posterior parameter of the second level latent \( \tilde{y}_2 \), and the ELBO to maximize as \( L(y_1, y_2) \). The posterior \( q(\tilde{y}_1, \tilde{y}_2|x) \) factorizes as \( q(\tilde{y}_1|x)q(\tilde{y}_2|y_1, x) \), which means that \( y_2 \) depends on \( y_1 \). Given \( y_1 \) is fixed, we can directly optimize \( y_2 \) by gradient ascent. However, it requires some tricks...
to optimize $y_1$. The intuition is, we do not want to find a $y_1$ that maximizes $\mathcal{L}(y_1, y_2)$ given a fixed $y_2$. Instead, we want to find a $y_1$, whose $\max_{y_1} \mathcal{L}(y_1, y_2)$ is maximum. This translates to the optimization problem as:

$$y_1 \leftarrow \arg \max_{y_1} \mathcal{L}(y_1, y_2^*(y_1)),$$

where $y_2^*(y_1) \leftarrow \arg \max_{y_2} \mathcal{L}(y_1, y_2)$.

(19)

In fact, Eq. 19 is a variant of setup in back-propagating through gradient ascent (Samuel & Tappen 2009; Domke 2012). The difference is, our $y_1$ also contributes directly to optimization target $\mathcal{L}(y_1, y_2)$. From this perspective, Eq. 19 is also closely connected to Kim et al. (2018), if we treat $y_1$ as the amortized encoder parameter and $y_2$ as latent.

And as SAVI on 1-level latent (Kim et al. 2018), we need to solve Eq. 19 using gradient ascent. Specifically, denote $\alpha$ as learning rate, $K$ as the total gradient ascent steps, $y_1^{k_1}$ as the $y_1$ after $k_1$ step update, $y_2^{k_2}$ as the $y_2$ after $k_2$ step update, and $f(\cdot)$ as FAVI initializing posterior parameters $y_0^1, y_0^2$. The optimization problem as Eq. 19 translates into the update rule as:

$$y_1^{k_1+1} \leftarrow y_1^{k_1} + \alpha \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_1^{k_1}},$$

$$y_2^{k_2+1} \leftarrow y_2^{k_2} + \alpha \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}},$$

(20)

To solve Eq. 20, we note that although $\frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}}$ can be directly computed, $\frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_1^{k_1}}$ is not straightforward. Let’s consider a simple example when the gradient ascent step $K = 1$:

- First, we initialize $y_1, y_2$ by FAVI $y_0^1 \leftarrow \text{FAVI}(x), y_0^2 \leftarrow \text{FAVI}(x, y_0^1)$.

- Next, we optimize $y_2$ by one step gradient ascent as $y_2^1 \leftarrow y_2^0 + \alpha \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}$. And this is where back-prop through gradient ascent (Samuel & Tappen 2009; Domke 2012) works:

$$\frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0} = \frac{\partial y_2^0}{\partial y_2^0} \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0} = (I + \alpha \frac{\partial^2 \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}) \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}.$$

- Next, we optimize $y_1$ to maximize $\mathcal{L}(y_0^1, y_2^1)$, and the gradient is

$$\frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_1^0} = \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_1^0} + \frac{\partial y_2^0}{\partial y_2^0} \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}.$$

With FAVI relationship, we naturally have $\frac{\partial y_1^0}{\partial y_0^1}$. And we need to evaluate is $\frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}$. And this is where back-prop through gradient ascent (Samuel & Tappen 2009; Domke 2012) works:

$$\frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0} = \frac{\partial y_2^0}{\partial y_2^0} \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0} = \frac{\partial^2 \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0} = \frac{\partial^2 \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}.$$

- Again, $\frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}$ is known.

- By now, we have collect all parts to solve $\frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}$, and we can finally update $y_1^0$ as $y_1^0 \leftarrow y_1^0 + \alpha \frac{\partial \mathcal{L}(y_1^0, y_2^0)}{\partial y_2^0}$.

For $K > 1$, we can extend the example and implement Eq. 20 as Alg. 2. Specifically, we first initialize $y_0^1$ from FAVI. Then we conduct gradient ascent on $y_1$ with gradient $\frac{\partial \mathcal{L}(y_1^k, y_2^k)}{\partial y_1^k}$ computed from the procedure grad-2-level$(x, y_1^k)$. And each time grad-2-level$(x, y_1^k)$ is evaluated, $y_2$ goes through a re-initialization and $K$ steps of gradient ascent. The above procedure corresponds to Eq. 20. The key of Alg. 2 is the evaluation of gradient $\frac{\partial \mathcal{L}(a^k, b^k)}{\partial a^k}$.

Formally, we have:

**Theorem 3.3.** After grad-2-level$(x, y_1^k)$ of Alg. 2 executes, we have the return value $\frac{\partial \mathcal{L}(y_1^k, y_2^k)}{\partial y_1^k} = y_1^k$. (See proof in Appendix B)

**Algorithm 2** Proposed Accurate SAVI on 2-level Latent

**procedure** solve-2-level$(x)$

initialize $y_0^1 \leftarrow f_\phi(x)$ from FAVI.

for $k_1 = 0$ to $K - 1$ do

$$\frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_1^{k_1}} = \text{grad-2-level}(x, y_1^{k_1}),$$

$$y_1^{k_1+1} \leftarrow y_1^{k_1} + \alpha \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_1^{k_1}}.$$

end for

return $y_1^K, y_2^K$

**procedure** grad-2-level$(x, y_1^{k_1})$

initialize $y_0^1 \leftarrow f_\phi(x, y_1^{k_1})$ from FAVI.

for $k_2 = 0$ to $K - 1$ do

$$\frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}} = \text{grad-2-level}(x, y_1^{k_1}),$$

$$y_2^{k_2+1} \leftarrow y_2^{k_2} + \alpha \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}}.$$

end for

initialize $y_1^0 \leftarrow f_\phi(x, y_1^{k_1}), y_2^K \leftarrow \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^K)}{\partial y_2^K}$

for $k_2 = K - 1$ to $0$ do

$$\frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}} = \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}} - \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}}.$$

$$y_1^{k_1} \leftarrow y_1^{k_1} + \alpha \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}}.$$

$$y_2^{k_2} \leftarrow y_2^{k_2} + \alpha \frac{\partial \mathcal{L}(y_1^{k_1}, y_2^{k_2})}{\partial y_2^{k_2}}.$$

end for

$$\frac{\partial \mathcal{L}(y_1^{k_1}, y_2^K)}{\partial y_1^{k_1}} = y_1^k.$$

**return**

**Accurate SAVI on DAG Latent** Then, we extend the result on 2-level latent to general non-factorized latent with dependency described by DAG. This DAG is the computational graph during network inference, and it is also the directed graphical model (DGM) (Koller & Friedman 2009) defining the factorization of latent variables during inference. This extension is necessary for SAVI on latent with complicated dependency (e.g. bit allocation of NVC).
Similar to the 2-level latent setup, we consider performing SAVI on $N$ variational posterior parameter $y_1, \ldots, y_N$ with their dependency defined by a computational graph $G$, i.e., their corresponding latent variable $\hat{y}_1, \ldots, \hat{y}_N$’s posterior distribution factorizes as $G$. Specifically, we denote $y_j \in C(y_i), y_i \in P(a_j)$ if an edge exists from $y_j$ to $y_i$. This indicates that $\hat{y}_j$ conditions on $\hat{y}_i$. Without loss of generality, we assume $y_1, \ldots, y_N$ is sorted in topological order. This means that if $y_j \in C(y_i), y_i \in P(y_j)$, then $i < j$. Each latent is optimized by $K$-step gradient ascent, and $y_{K_i}^k$ denotes the latent $y_i$ after $k_i$ steps of update. Then, similar to 2-level latent, we can solve this problem by recursively applying back-propagating through gradient ascent (Samuel & Tappen [2009], Domke [2012]) to obtain Alg. 3.

Specifically, we add a fake latent $y_0$ to the front of all $y$s. Its children are all the $y$s with 0 in-degree. Then, we can solve the SAVI on $y_1, \ldots, y_N$ using gradient ascent by executing the procedure grad-dag$(x, y_0^0, \ldots, y_k^k)$ in Alg. 3 recursively. Inside procedure grad-dag$(x, y_0^0, \ldots, y_k^k)$, the gradient to update $y_i$ relies on the convergence of its children $y_j \in C(y_i)$, which is implemented by the recursive depth-first search (DFS) in line 11. And upon the completion of procedure grad-dag$(x, y_0^0, \ldots, y_k^k)$, all the latent converges to $y_0^k, \ldots, y_N^k$. Similar to the 2-level latent case, the key of Alg. 3 is the evaluation of gradient $dL(y_0^0, \ldots, y_k^k, \ldots, y_N^k)/dy_i^k$. Formally, we have:

**Theorem 3.4.** After the procedure grad-dag$(x, y_0^0, \ldots, y_k^k)$ in Alg. 3 executes, we have the return value $dL(y_0^0, \ldots, y_k^k, \ldots, y_N^k)/dy_i^k = \frac{\partial}{\partial y_i^k}$. (See proof in Appendix [B]).

### 4. Complexity Reduction

An evident problem of the accurate SAVI (Sec. 3.4) is the temporal complexity. Given the frame number $N$ and gradient ascent step $K$, Alg. 3 has temporal complexity of $\Theta(KN)$. NVC with GoP size 10 has approximately $N = 20$ latent, and the SAVI on neural image compression (Yang et al. 2020) (Gao et al. 2022) takes around $K = 2000$ steps to converge. For bit allocation, the complexity of Alg. 3 is approximately $2000^{20}$, which is intractable.

#### 4.1. Temporal Complexity Reduction

Therefore, we provide an approximation to the SAVI on DAG. The general idea is that, the SAVI on DAG (Alg. 3) satisfies both requirement on gradient signal described in Sec. 3.3. We can not make it tractable without breaking them. Thus, we break one of them for tractable complexity, while maintaining good performance. Specifically, We consider the approximation as:

$$
\frac{dL(y_0^0, \ldots, y_k^k, \ldots, y_N^k)/dy_i^k}{dy_i^k} \approx \frac{dL(y_0^0, \ldots, y_k^k, y_i^k)}{dy_i^k},
$$

which obeys the requirement 1 in Sec. 3.3 while breaks the requirement 2. Based on Eq. (21) we design an approximation of SAVI on DAG. Specifically, with the approximation in Eq. (21) the recurrent gradient computation in Alg. 3 becomes unnecessary as the right hand side of Eq. (21) does not require $y_i^K$. However, to maintain the dependency of latent, we still need to ensure that the children node $y_j \in C(y_i)$ are re-initialized by FAVI every-time when $y_i$ is updated. Therefore, we traverse the graph in topological order, keep the children node $y_j$ untouched until all its parent node $y_i \in P(y_j)$’s gradient ascent is completed. And the resulting approximate SAVI is Alg. 4. Its temporal complexity is $\Theta(KN)$.
100 (Hu et al., 2022). Therefore, it is important to reduce the likelihood range. We evaluate the negative log likelihood (NLL) lowerbound for gradient evaluation from the whole GoP to frames: $y_i$ for $i = 0, 1, \ldots, K$. As the proposed SA VI without approximation (Proposed-Approx, Sec. 4.1) and proposed SA VI with scalable approximation (Proposed-Scalable, Sec. 4.2) on 3 NVC baselines: DVC (Li et al., 2019), DCVC (Li et al., 2021) and HSTEM (Li et al., 2022a). For all 3 baseline methods, we adopt the official pre-trained models. As DVC and DCVC do not provide I frame model, we adopt Cheng et al. (2020) with pre-trained models provided by Begaint et al. (2020). Following baselines, we adopt HEVC Common Testing Condition (CTC) (Bossen et al., 2013) and UVG dataset (Mercat et al., 2020). And the GoP size is set to 10 for HEVC CTC and 12 for UVG dataset. The R-D performance is measured by Bjontegaard-Bitrate (BD-BR) and BD-PSNR (Bjontegaard, 2001). For the proposed approach, we evaluate the approximated SAVI (Alg. 4) and its scalable version with $C = 2$. We adopt Adam (Kingma & Ba, 2014) optimizer with $lr = 1 \times 10^{-3}$ to optimize $y_{1:N}$ for $K = 2000$ iterations. For other bit allocation methods, we select a traditional $\lambda$-domain approach (Li et al., 2016), a recent $\lambda$-domain approach (Li et al., 2020) and online encoder update (OEU) (Lu et al., 2020a). More details are presented in Appendix C.

4.2. Spatial Complexity Reduction

Despite the the approximated SAVI on DAG reduce the temporal complexity from $\Theta(K^N)$ to $\Theta(N)$, the spatial complexity remains $\Theta(N)$. Though most of NVC approaches adopt a small GoP size of 10-12 (Lu et al., 2019; Li et al., 2021), there are emerging approaches extending GoP size to 100 (Hu et al., 2022). Therefore, it is important to reduce the spatial complexity to constant for scalability.

Specifically, our approximated SAVI on DAG uses the gradient of GoP-level likelihood $L$, which takes $\Theta(N)$ memory. We empirically find that we can reduce the likelihood range for gradient evaluation from the whole GoP to $C$ future frames:

$$\frac{\partial L}{\partial y_{k}^i} = \sum_{j=1}^{N} \frac{\partial L}{\partial y_{j}^i} \approx \sum_{j=1}^{N} \frac{\partial L}{\partial y_{j}^i} \tag{22}$$

where $C$ is a constant. Then, the spatial complexity is reduced to $\Theta(C)$, which is constant to GoP size $N$.

5. Experimental Results

5.1. Evaluation on Density Estimation

As the proposed SAVI without approximation (Proposed-Accurate, Sec. 4.1) is intractable for NVC, we evaluate it on small density estimation tasks. Specifically, we consider a 2-level VAE with inference model $x \rightarrow \tilde{y}_1 \rightarrow \tilde{y}_2$, where $x$ is observed data, $\tilde{y}_1$ is the first level latent with dimension 100 and $\tilde{y}_2$ is the second level latent with dimension 50. We adopt the same 2-level VAE architecture as Burda et al. (2015). And the dataset we use is MNIST (LeCun et al., 1998). More details are provided in Appendix C.

We evaluate the negative log likelihood (NLL) lowerbound of FAVI, Original SAVI (Alg 1), proposed SAVI without approximation (Proposed-Accurate, Sec. 4.1) and proposed SAVI with approximation (Proposed-Approx, Sec. 4.1). Tab. 2 shows that our Proposed-Accurate achieves a lower NLL ($\leq 91.5482$) than original SAVI ($\leq 91.5530$). And our Proposed-Approx ($\leq 91.5486$) is only marginally out-performed by Proposed-Accurate, which indicates that this approximation does not harm performance significantly. As the mean NLL difference between Proposed-Accurate and Proposed-Approx is small, we additionally perform pairwise t-test between methods and report p-values. And the results suggest that the difference between methods is significant ($p \leq 0.001$).

5.2. Evaluation on Bit Allocation

Experiment Setup We evaluate our bit allocation method based on 3 NVC baselines: DVC (Lu et al., 2019), DCVC (Li et al., 2021) and HSTEM (Li et al., 2022a). For all 3 baseline methods, we adopt the official pre-trained models. As DVC and DCVC do not provide I frame model, we adopt Cheng et al. (2020) with pre-trained models provided by Begaint et al. (2020). Following baselines, we adopt HEVC Common Testing Condition (CTC) (Bossen et al., 2013) and UVG dataset (Mercat et al., 2020). And the GoP size is set to 10 for HEVC CTC and 12 for UVG dataset. The R-D performance is measured by Bjontegaard-Bitrate (BD-BR) and BD-PSNR (Bjontegaard, 2001). For the proposed approach, we evaluate the approximated SAVI (Alg. 4) and its scalable version with $C = 2$. We adopt Adam (Kingma & Ba, 2014) optimizer with $lr = 1 \times 10^{-3}$ to optimize $y_{1:N}$ for $K = 2000$ iterations. For other bit allocation methods, we select a traditional $\lambda$-domain approach (Li et al., 2016), a recent $\lambda$-domain approach (Li et al., 2020) and online encoder update (OEU) (Lu et al., 2020a). More details are presented in Appendix C.

Main Results We extensively evaluate the R-D performance of proposed SAVI with approximation (Proposed-Approx, Sec. 4.1) and proposed SAVI with scalable approximation (Proposed-Scalable, Sec. 4.2) on 3 baselines and 5 datasets. As Tab. 1 shows, for DVC (Lu et al., 2019) and DCVC (Li et al., 2021), our Proposed-Approx and Proposed-Scalable outperform all other bit allocation methods by large margin. It shows that the current best bit allocation method, OEU (Lu et al., 2020a), still has a room of 0.5 dB BD-PSNR to improve compared with our result. For HSTEM (Li et al., 2022a) with large model, we only evaluate our Proposed-Scalable. As neither OEU (Lu et al., 2020a) nor Proposed-Approx is able to run on HSTEM within 80G RAM limit. Moreover, as HSTEM has a bit allocation module, the effect of bit allocation is not as evident as DVC and DCVC. However, our bit allocation still brings a significant improvement of more than 0.4 dB PSNR over HSTEM.

Ablation Study We evaluate the R-D performance, encoding time and memory consumption of different methods with DVC (Lu et al., 2019) on HEVC Class D. As Tab. 3 and Fig. 2 upper-left show, the Proposed-Approx (-35.86%) significantly outperforms the original SAVI (-14.76%) in terms of BD-BR. Furthermore, it significantly

Algorithm 4 Proposed Approximated SAVI

procedure solve-approx($x$)
    for $i = 1 \rightarrow N$
do
        initialize $y_{1}^{i}, \ldots, y_{K}^{i} \leftarrow f_{\theta}(x, y_{K}^{i})$ from FAVI.
    for $k = 0 \rightarrow K - 1$
do
        $\frac{\partial L}{\partial y_{k}^{i}} \approx \sum_{j=1}^{N} \frac{\partial L}{\partial y_{j}^{i}} \tag{22}$
        $y_{k+1}^{i} \leftarrow y_{k}^{i} + \alpha \frac{\partial L}{\partial y_{k}^{i}}$
    end for
end for
return $y_{1}^{1}, \ldots, y_{K}^{N}$

Bit Allocation using Optimization
Table 1. The BD-BR and BD-PSNR of our approach compared with baselines (w/o bit allocation) and other bit allocation approaches. ∗ The data comes from Li et al. (2022b). We mark some data with '-' as they are not reported by Li et al. (2022b).

Table 2. The NLL result on density estimation.

Table 3. The R-D performance and the encoder resource consumption of different approaches. Encode time is per-frame and measured with AMD EPYC 7742 CPU and Nvidia A100 GPU.

Table 4. The R-D performance and the encoding time consumption of different gradient ascent step and learning rate. We assume that the temporal complexity is approximately linear to steps.

outperforms all other bit allocation approaches. On the other hand, our Proposed-Scalable effectively decreases encoding time (from 528.3s to 74.6s). However, its R-D performance (-29.81%) remains superior than all other bit allocation approaches by large margin. On the other hand, SAVI based approaches improve R-D performance not only by bit allocation, but also by reducing the amortization gap and discretization gap (Yang et al., 2020b). To investigate the net improvement by bit allocation, we perform (Yang et al., 2020b) in a per-frame manner. And the resulting method (Original SAVI (per-frame)) improves the R-D performance by 20%. This means that the net enhancement brought by bit allocation is around 15%. Furthermore, as shown in Fig.2<sup>upper-right</sup>, the memory requirement of our Proposed-Scalable is constant to GoP size. While for OEU (Lu et al., 2020a), original SAVI and Proposed-Approx, the RAM is linear to GoP size. Despite the encoding time and RAM of our approach is higher than baseline, the decoding remains the same. Furthermore, compared with other encoder for R-D performance benchmark purpose (VTM 13.2), the encoding time and memory of our approach is reasonable.

For both Proposed-Approx and Proposed-Scalable, it is possible to achieve speed-performance trade-off by adjusting gradient ascent steps and learning rate. In Tab.4 we can see that reducing number of gradient ascent steps linearly reduce the encoding time while maintain competitive R-D performance. Specifically, Proposed-Scalable with 500 steps and $1 \times 10^{-3}$ lr achieves similar BD-BR (-18.99 vs -18.97%) and encoding time (18.6 vs 15.2s) as OEU (Lu et al., 2020a), with less than half RAM requirement (2.12 vs 5.83GB).

Analysis To better understand our bit allocation methods, we show the per-frame bpp and PSNR before and after bit allocation in Fig.2<sup>lower-left</sup> and Fig.2<sup>lower-right</sup>. It can be
Bit Allocation using Optimization

for all data, which leads to the amortization gap (Cremer et al., 2018). SAVI reduces this gap by optimizing the variational posterior parameter after initializing it with inference network. It adopts back-propagating through gradient ascent (Domke, 2012) to evaluate the gradient of amortized parameters. When applying SAVI to practical neural codec, researchers abandon the nested parameter update for efficiency. Prior works (Djelouah & Schroers, 2019; Yang et al., 2020b; Zhao et al., 2021; Gao et al., 2022) adopt SAVI to boost R-D performance and achieve variable bitrate in image compression.

7. Discussion & Conclusion

Despite the proposed approach has reasonable complexity as a benchmark, its encoding speed remains far from real-time. We will continue to investigate faster approximation. (See more in Appendix.I). Another limitation is that current evaluation is limited to NVC with P-frames.

To conclude, we show that the SAVI using GoP-level likelihood is equivalent to optimal bit allocation for NVC. We extend the original SAVI to non-factorized latent by back-propagating through gradient ascent and propose a feasible approximation to make it tractable for bit allocation. Experimental results show that current bit allocation methods still have a room of 0.5 dB PSNR to improve, compared with our optimal result.

Acknowledgements

This work was supported by Xiaomi AI Innovation Research under grant No.202-422-002.

References


Bit Allocation using Optimization


Bit Allocation using Optimization


A. Details of $\lambda$-domain Bit Allocation.

The $\lambda$-domain bit allocation (Li et al., 2016) is a well-acknowledged work in bit allocation. It is adopted in the official reference software of H.265 as the default bit allocation algorithm. In Sec. 2.2, we briefly list the main result of Li et al. (2016) without much derivation. In this section, we dive into the detail of it.

To derive Eq. 10, we note that we can expand the first optimal condition Eq. 9 as:

$$
\sum_{j=1}^{N} \frac{dR_j}{dR_i} \frac{dR_i}{d\lambda_i} + \lambda_0^T \sum_{j=1}^{N} \frac{dD_j}{d\lambda_i} \frac{dD_i}{d\lambda_i} = 0. \quad (23)
$$

Then immediately we notice that the $dR_j/dR_i, dD_j/dD_i$ terms can be represented as the empirical model in Eq. 9. After inserting Eq. 9 we have:

$$
I \frac{dR_i}{d\lambda_i} + \lambda_0^T \omega_i I \frac{dD_i}{d\lambda_i} = 0. \quad (24)
$$

And now we can obtain Eq. 10 by omitting the identity matrix $I$.

To derive Eq. 11, we multiply both side of Eq. 10 by $d\lambda_i/dD_i$, and take the second optimal condition Eq. 8 into it:

$$
\frac{dR_i}{d\lambda_i} \frac{d\lambda_i}{dD_i} + \omega_i \lambda_0^T I \frac{dD_i}{d\lambda_i} = \frac{dR_i}{dD_i} + \omega_i \lambda_0^T I,
$$

which implies that we have:

$$
\lambda_i' = (I + \sum_{j=1}^{N} \frac{dD_j}{dD_i})^T \lambda_0/(1 + \sum_{j=1}^{N} \frac{dR_j}{dR_i}),
$$

(25)

and find out that we can obtain the result of $\lambda$-domain bit allocation. This implies that our SAVI-based bit allocation is the $\lambda$-domain bit allocation with a different rate & quality dependency model, which is accurately defined via gradient of NVC, instead of empirical.

B. Proof of Main Results.

In Appendix B, we provide proof to our main theoretical results.

**Theorem 3.1.** The SAVI using $GoP$-level likelihood is equivalent to bit allocation with:

$$
\lambda_i' = (1 + \sum_{j=1}^{N} \frac{dD_j}{dD_i})^T \lambda_0/(1 + \sum_{j=1}^{N} \frac{dR_j}{dR_i}),
$$

(14)

where $dD_j/dD_i, dR_j/dR_i$ can be computed numerically through the gradient of NVC model.

**Proof.** To find $\lambda_i'$, we expand the definition of $\lambda_i'$ in Eq. 13 as:

$$
\frac{d(R_i + \lambda_i'^T D_i)}{dy_i} = \frac{d(R_i + \lambda_i^T D_i)}{dy_i} + \sum_{j=1}^{N} \frac{d(D_j)}{dy_i} \lambda_i^T (1 + \sum_{j=1}^{N} \frac{d(D_j)}{dy_i} \frac{D_j}{D_i})/dy_i,
$$

(27)

$$
\lambda_i' = (1 + \sum_{j=1}^{N} \frac{dD_j}{dD_i})^T \lambda_0/(1 + \sum_{j=1}^{N} \frac{dR_j}{dR_i}),
$$

(29)

We note that $dR_j/dy_i$ and $dD_j/dy_i$ can be evaluated directly on a computational graph. On the other hand, $dR_i/dD_i$ and $dD_i/dy_i$ can also be evaluated directly on a computational graph. Then, we can numerically obtain the reverse direction gradient $dy_i/R_i$ and $dy_i/D_i$ by taking reciprocal of each element of the Jacobian:

$$
(dy_i/R_i)^{mn} = 1/(dy_i)^{mn}, \quad (dy_i/D_i)^{mn} = 1/(dy_i)^{mn},
$$

(29)

where $m, n$ are the location index of the Jacobian matrix. Now all the values in Eq. 28 are solved, we can numerically evaluate the rate & quality dependency model.
**Theorem 3.2.** The equivalent bit allocation map $\lambda_i^*$ is the solution to the optimal bit allocation problem in Eq. 3. In other words, we have:

$$\lambda_i' = \lambda_i^*.$$  \hfill (15)

**Proof.** First, we will solve $\lambda_i^*$ of optimal bit allocation in Eq. 3 analytically with the precise R-D model defined by the computational graph. And then we will prove Theorem 3.2 by showing that $\lambda_i^* = \lambda_i$.

To solve $\lambda_i^*$, we expand $dR_j/d\lambda_i$ and $dD_j/d\lambda_i$ as:

$$\frac{dR_j}{d\lambda_i} = \frac{dR_j}{d\lambda_i}, \quad \frac{dD_j}{d\lambda_i} = \frac{dD_j}{d\lambda_i}. \hfill (30)$$

Then immediately, we have:

$$\lambda_i^* = (1 + \sum_{j=i+1}^N \frac{dD_j}{d\lambda_i}) \lambda_i^*(1 + \sum_{j=i+1}^N \frac{dD_j}{d\lambda_i}).$$

Further, multiply each side of Eq. 31 by $d\lambda_i/dD_i$ and take Eq. 31 into it, we have:

$$-\lambda_i^T(1 + \sum_{j=i+1}^N \frac{dD_j}{d\lambda_i}) + \lambda_i^*(1 + \sum_{j=i+1}^N \frac{dD_j}{d\lambda_i}) = 0.$$ \hfill (32)

Then immediately, we have:

$$\lambda_i^* = (1 + \sum_{j=i+1}^N \frac{dD_j}{d\lambda_i}) \lambda_i^*/(1 + \sum_{j=i+1}^N \frac{dD_j}{d\lambda_i}) = \lambda_i' = \lambda_i^*,$$ \hfill (33)

which proves Theorem 3.2.

**Theorem 3.3.** After grad-2-level($x_i, y_i^k$) of Alg. 2 executes, we have the return value $dL(y_i^k, y_i^k)/dy_i^k = y_i$.

**Proof.** This proof extends the proof of Theorem 1 in Domke (2012). Note that our algorithm is different from Samuel & Tappen (2009), Domke (2012), Kim et al. (2013) as our high level parameter $y_i$ not only generate low level parameter $y_k$, but also directly contributes to optimization target (See Fig. 3).

As the computational graph in Fig. 3 shows, we can expand $dL(y_i^k, y_i^k)/dy_i^k$ as:

$$\frac{dL(y_i^k, y_i^k)}{dy_i^k} = \frac{\partial L(y_i^k, y_i^k)}{\partial y_i^k} + \sum_{k=0}^K \frac{\partial y_i^k}{\partial y_i^k} \frac{dL(y_i^k, y_i^k)}{dy_i^k}.$$ \hfill (34)

To solve Eq. 34 we first note that $\partial L(y_i^k, y_i^k)/\partial y_i^k$, $dL(y_i^k, y_i^k)/dy_i^k$ and $\partial y_i^k/\partial y_i^k$ is naturally known.

Then, by taking partial derivative of the update rule of gradient ascent $y_i^{k+1} := y_i^k + \alpha dL(y_i^k, y_i^k)/dy_i^k$ with regard to $y_i^{k+1}$, we have:

$$\frac{\partial y_i^{k+1}}{\partial y_i^k} = I + \alpha \frac{dL(y_i^k, y_i^k)}{\partial y_i^k}, \hfill (35)$$

$$\frac{\partial y_i^{k+1}}{\partial y_i^k} = \alpha \frac{dL(y_i^k, y_i^k)}{\partial y_i^k}. \hfill (36)$$

Note that Eq. 36 is the partial derivative $\partial y_i^{k+1}/\partial y_i^k$ instead of total derivative $dy_i^{k+1}/dy_i^k$, whose value is $(\partial y_i^{k+1}/\partial y_i^k)(dy_i^{k+1}/dy_i^k) + \partial y_i^{k+1}/\partial y_i^k$. And those second order terms can either be directly evaluated or approximated via finite difference as Eq. 38.

As Eq. 38 already solves the first term on the right hand side of Eq. 34, the remaining issue is $dL(y_i^k, y_i^k)/dy_i^k$. To solve this term, we expand it recursively as:

$$\frac{dL(y_i^k, y_i^k)}{dy_i^k} = \frac{\partial y_i^{k+1}}{\partial y_i^k} \frac{dL(y_i^k, y_i^k)}{dy_i^k}.$$ \hfill (37)

Then, we have solved all parts that is required to evaluate $L(y_i^k, y_i^k)/dy_i^k$. The above solving process can be described by the procedure grad-2-level($x_i, y_i^k$) of Alg. 2.

Specifically, the iterative update of $y_i^k$ corresponds to recursively expanding Eq. 37 with Eq. 35 and the iterative update of $y_i^k$ corresponds to recursively expanding Eq. 36 with Eq. 36 and Eq. 37. Upon the return of grad-2-level($x_i, y_i^k$) of Alg. 2, we have $y_i^k = dL(y_i^k, y_i^k)/dy_i^k$.

The complexity of the Hessian-vector product in Alg. 2 may be reduced using finite difference following (Domke 2012).
We highlight several key differences between Alg. 3 and Alg. 2: we expand each child node as Eq. 37 from the proof of Theorem 3.3. Then, the  

\[ \frac{\partial^2 L(y_{i1}, y_{i2})}{\partial y_{i1} \partial y_{i2}} \]  

be conducted trivially only if \( C(y_j) \) is empty, otherwise the gradient has to be evaluated recursively using Eq. 39. And this part corresponds to the recursive call in line 11 of Alg. 3. 

And the other part of Alg. 3 is the same as Alg. 2. So the rest of the proof follows Theorem 3.3. Similarly, the Hessian-vector product can be approximated as Eq. 38. However, this does not save Alg. 3 from an overall complexity of \( O(K^N) \).

To better understand Alg. 3 we provide an example of the full procedure of its execution in Fig. 7. The setup is as Fig. 7(0): we have \( N = 3 \) latent \( y_1, y_2, y_3 \) and gradient ascent step \( K = 2 \), connected by a DAG shown in the figure. □

C. Implementation Details

C.1. Implementation Details of Density Estimation

For density estimation experiment, we follow the training details of Burda et al. (2015). We adopt Adam (Kingma & Ba, 2014) optimizer with \( \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e^{-4} \) and batch-size 20. We adopt the same learning rate scheduler as (Burda et al., 2015) and train 3280 epochs for total. All likelihood results in Tab. 2 are evidence lowerbound. And the binarization of MNIST dataset follows (Salakhutdinov & Murray, 2008).

C.2. Implementation Details of Bit Allocation

In the main text and analysis, we use \( y_i \) to abstractly represent all the latent of frame \( i \). While the practical implementation is more complicated. In fact, the actual latent is divided into 8 parts: the first level of residual latent \( z_{i1}^{res} \), the second level of residual latent \( z_{i2}^{res} \), the first level of optical flow latent \( y_{i1}^{mv} \), and the second level of optical flow latent \( z_{i2}^{mv} \). In addition to those 4 latent, we also include quantization stepsizes \( \Delta_{i1}^{res}, \Delta_{i2}^{res}, \Delta_{i1}^{mv}, \Delta_{i2}^{mv} \) as Choi et al. (2019) for DVC (Lu et al., 2019) and DCVC (Li et al., 2021). For HSTEM (Li et al., 2022a), those quantization parameters are predicted from the latent \( z_i \), so we have not separately optimize them. This is shown in the overall framework diagram as Fig. 4. To re-parameterize the quantization, we adopt Stochastic Gumbel Annealing with the hyper-parameters as Yang et al. (2020b).

D. More Experimental Results

D.1. R-D Performance

Fig. 5 shows the R-D curves of proposed approach, other bit allocation approach (Lu et al., 2020a) and baselines (Lu et al., 2019; Li et al., 2021, 2022a), from which we observe that proposed approach has stable improvement upon three...
baselines and five datasets (HEVC B/C/D/E, UVG).

In addition to baselines and bit allocation results, we also show the R-D performance of traditional codecs. For H.265, we test x265 encoder with veryslow preset. For H.266, we test VTM 13.2 with lowdelay P preset. The command lines for x265 and VTM are as follows:

```bash
ffmpeg -y -pix_fmt yuv420p -s HxW -r 50 -i src_yuv -vframes frame_cnt -c:v libx265 -preset veryslow -x265-params "qp=qp:
keyint=gop_size" output_bin

EncoderApp -c encoder_lowdelay_P_vtm.cfg --Qp=qp --InputFile=src_yuv --BitstreamFile=output_bin
--DecodingRefreshType=2 --InputBitDepth=8 --OutputBitDepth=8 --InputChromaFormat=420 --Level=6.2 --FrameRate=50 --FramesToBeEncoded=frame_cnt --SourceWidth=W --SourceHeight=H
```

And for x265, we test qp={38, 34, 32, 28, 24}. For VTM 13.2, we test qp={34, 30, 26, 24, 22}.

As Fig. 6 shows, with our bit allocation, the DCVC (Li et al., 2021) outperforms the latest traditional codec standard (VTM 13.2). The HSTEM baseline (Li et al., 2022a) already outperforms VTM 13.2, and our bit allocation makes this advantage more obvious.

D.2. Qualitative Results

In Fig. 8, Fig. 9, Fig. 10 and Fig. 11, we present the qualitative result of our approach compared with the baseline approach on HEVC Class D. We note that compared with the reconstruction frame of baseline approach, the reconstruction frame of our proposed approach preserves significantly more details with lower bitrate, and looks much more similar to the original frame.

E. More on Online Encoder Update (OEU)

E.1. Why OEU is a Frame-level Bit Allocation

The online encoder update (OEU) of Lu et al. (2020a) is proposed to resolve the error propagation. It fits into the line of works on encoder over-fitting (Cremer et al., 2018; van Rozendaal et al., 2021; Zou et al., 2020) and does not fit into the SAVI framework. However, it can be seen as an intermediate point between fully-amortized variational inference (FAVI) and SAVI: FAVI uses the same inference model parameter $\phi$ for all frames; SAVI performs stochastic variational inference on variational posterior parameter $y_i$ per frame, which is the finest-granularity finetune possible for latent variable model; And Lu et al. (2020a) finetune inference model parameter $\phi_i$ per-frame. It is less computational expensive but also performs worse compared with our method based on SAVI (See Tab. 5). However, it is more computational expensive and performs better compared with FAVI (or NVC without bit allocation).

Although OEU does not fit into SAVI framework, still it can be written as Eq. 40 which is very similar to our SAVI-based implicit bit allocation with $\Delta_i$ removed and optimization target changed from $y_i, \Delta_i$ to encoder parameter $\phi_i$.

$$
\phi_{i,K}^* \leftarrow \arg \max_{\phi_{i,K}} \mathcal{L}
$$

(40)
Table 5. A comparison between DVC (FA VI, w/o bit allocation), OEU (Lu et al., 2020a) and our proposed bit allocation (SA VI) approach. N is the GoP size, $|\phi|$ is the number of encoder parameters, $H \times W$ is the number of pixels, $K$ is number of iteration.

This similarity also implies that Lu et al. (2020a) is optimized towards the overall bit allocation target. Thus, it is also an optimal bit allocation. However, the encoder parameter $\phi_i$ is only sensitive to the location of a frame inside a GoP. It can not sense the pixel location and pixel-level reference structure. So the bit allocation of Lu et al. (2020a) is limited to frame-level instead of pixel-level like ours.

Despite it is not pixel-level optimal bit allocation, OEU (Lu et al., 2020a) is indeed the true pioneer of bit allocation in NVC. However, the authors of Lu et al. (2020a) have not mentioned the concept of bit allocation in the original paper, which makes it a secret pioneer of bit allocation for NVC until now.

E.2. Why OEU is not an Error Propagation Aware Strategy

One problem of NVC is that the frame quality drops rapidly with the frame index $t$ inside the GoP. Many works call this phenomena error propagation (Lu et al., 2020a; Sun et al., 2021; Sheng et al., 2021), including the OEU. And the error propagation aware technique is subtly related to bit allocation.

From the old school bitrate control’s perspective (Tagliasacchi et al., 2008), the target of avoiding error propagation is $\minVar$, which means that we want to minimize the quality fluctuation between frames. On the other hand, the target of bit allocation is $\minAvg$, which means that we want to minimize the average distortion under constrain of bitrate. And usually, those two targets are in odd with each other.

However, the results of a crippled error propagation aware method and an optimal bit allocation are similar: the degradation speed of quality-$t$ curve is decreased but not flattened. And thus, sometimes the result of an optimal bit allocation is likely to be recognised as an unsuccessful error propagation aware method. This similarity has disoriented Lu et al. (2020a) to falsely classify themselves into error propagation aware strategy instead of bit allocation.

As we empirically show in Fig. 5 lower-left, the quality-$t$ curves of both our proposed method and OEU are not horizontal. Instead, they are just less steep than the original DVC (Lu et al., 2019). This phenomena further verifies that OEU is essentially a bit allocation, instead of an error propagation aware strategy.
F. More Discussion

F.1. More Limitations

As we have stated in main text, the major limitation of our method is encoding complexity. Although the decoding time is not influenced, our method requires iterative gradient descent during encoding. Practically, our approach is limited to scenarios when R-D performance matters more than encoding time (e.g. content delivery network). Or it can also be used as a benchmark for research on faster bit allocation methods.

F.2. Ethics Statement

Improving the R-D performance of NVC has valuable social impact. First, better R-D performance means less resources required for video storage and transmission. Considering the amount of video produced everyday, it is beneficial to the environment if we could save those resources by improving codec. Second, the traditional codec requires dedicated hardware decoder for efficient decoding, while NVC could utilize the universal neural accelerator. The improvement of neural codec can be readily deployed without hardware upgradation.
Figure 6. The R-D performance of our approach compared with baselines (w/o bit allocation) and other bit allocation approach.
Figure 7. (0). The example setup. (1)- (27). The execution procedure. The number in the circle indicates the gradient step $k_i$ of each node $y_i$. The bold blue/red arrow indicates that the current node is under initialization/gradient ascent.
Figure 8. Qualitative results using BasketballPass of HEVC Class D. Top. Original frame. Middle. Baseline codec (DVC)'s reconstruction result with bpp = 0.148 and PSNR = 33.06dB. Bottom. Proposed method’s reconstruction result with bpp = 0.103 and PSNR = 34.91dB.
Figure 9. Qualitative results using BlowingBubbles of HEVC Class D. Top. Original frame. Middle. Baseline codec (DVC)’s reconstruction result with bpp = 0.206 and PSNR = 30.71dB. Bottom. Proposed method’s reconstruction result with bpp = 0.129 and PSNR = 32.34dB.
Figure 10. Qualitative results using BQSquare of HEVC Class D. Top. Original frame. Middle. Baseline codec (DVC)'s reconstruction result with bpp = 0.232 and PSNR = 28.72dB. Bottom. Proposed method’s reconstruction result with bpp = 0.128 and PSNR = 30.87dB.
Figure 11. Qualitative results using RaceHorses of HEVC Class D. Top. Original frame. Middle. Baseline codec (DVC)’s reconstruction result with bpp = 0.448 and PSNR = 30.48dB. Bottom. Proposed method’s reconstruction result with bpp = 0.379 and PSNR = 31.92dB.