CORRECTING THE SUB-OPTIMAL BIT ALLOCATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we investigate the problem of bit allocation in Neural Video Compression (NVC). First, we reveal that a recent bit allocation approach claimed to be optimal is, in fact, sub-optimal due to its implementation. Specifically, we find that its sub-optimality lies in the improper application of semi-amortized variational inference (SAVI) on latent with non-factorized variational posterior. Then, we show that the corrected version of SAVI on non-factorized latent requires recursively applying back-propagating through gradient ascent, based on which we derive the corrected optimal bit allocation algorithm. Due to the computational in-feasibility of the corrected bit allocation, we design an efficient approximation to make it tractable. Empirical results show that our proposed correction significantly improves the incorrect bit allocation in terms of R-D performance and bitrate error, and outperforms all other bit allocation methods by a large margin. The source code is provided in the supplementary material.

1 INTRODUCTION

Recently, bit allocation for Neural Video Compression (NVC) has drawn growing attention thanks to its great potential in boosting compression performance. Due to the frame reference structure in video coding, it is sub-optimal to use the same R-D (Rate-Distortion) trade-off parameter λ for all frames. In bit allocation task, bitrate is allocated to different frames/regions to minimize R-D cost $R + \lambda D$, where R is total bitrate, D is total distortion, and λ is the Lagrangian multiplier controlling R-D trade-off. Li et al. (2022) are the pioneer of bit allocation for NVC, who improve the empirical R-D (Rate-Distortion) model from traditional video codec (Li et al., 2014; 2016) and solve the per-frame Lagrangian multiplier λ. Other concurrent works adopt simple heuristics for coarse bit allocation (Cetin et al., 2022; Hu et al., 2022).

Most recently, BAO (Bit Allocation using Optimization) (Xu et al., 2022) proposes to formulate bit allocation as semi-amortized variational inference (SAVI) (Kim et al., 2018; Marino et al., 2018) and solves it by gradient-based optimization. Specifically, it directly optimizes the variational posterior parameter to be quantized and encoded by gradient ascent, aiming at maximizing the minus overall R-D cost, which is also the evident lowerbound (ELBO). BAO does not rely on any empirical R-D model and thus outperforms previous work. Further, BAO shows its optimality by proving its equivalence to bit allocation with precise R-D model.

In this paper, we first show that BAO (Xu et al., 2022) is in fact, sub-optimal due to its implementation. Specifically, we find that it abuses SAVI (Kim et al., 2018; Marino et al., 2018) on latent with non-factorized variational posterior, which brings incorrect gradient signal during optimization. To solve this problem, we first extend SAVI to non-factorized latent by back-propagating through gradient ascent (Domke, 2012). Then based on that, we correct the sub-optimal bit allocation in BAO to produce true optimal bit allocation for NVC. Furthermore, we propose a computational feasible approximation to such correct but intractable bit allocation method. And we show that our approximation outperforms the incorrect bit allocation (BAO) in terms of R-D performance and bitrate error, and performs better than all other bit allocation methods.

To summarize, our contributions are as follows:

- We demonstrate that a previously claimed optimal bit allocation method is actually sub-optimal. We find that its sub-optimality comes from the improper application of SAVI to non-factorized latent.

- We present the correct way to conduct SAVI on non-factorized latent by recursively applying back-propagation through gradient ascent. Based on this, we derive the corrected optimal bit allocation algorithm for NVC.
- Furthermore, we propose a computational efficient approximation of the optimal bit allocation to make it feasible. Our proposed approach improves the R-D performance and bitrate error over the incorrect bit allocation (BAO), and outperforms all other bit allocation methods for NVC.

2 PRELIMINARIES

2.1 NEURAL VIDEO COMPRESSION

The input of NVC is a GoP (Group of Picture) $\boldsymbol{x}_{1:T}$, where $\boldsymbol{x}_i \in R^{H\times W}$ is the i^{th} frame with $H \times W$ pixels, and T is the number of frame inside the GoP. Most of the works in NVC follow a latent variable model with temporal autoregressive relationship (Yang et al., 2020a). Specifically, to encode $\boldsymbol{x}_i$, we first extract the motion latent $\boldsymbol{w}_i = f_\phi^w(\boldsymbol{x}_i, \boldsymbol{x}'_i)$ from current frame $\boldsymbol{x}_i$ and previous reconstructed frame $\boldsymbol{x}'_{i-1}$, where $f_\phi^w(\cdot)$ is the motion encoder parameterized by ϕ[1]. Then, we encode the quantized latent $\tilde{\boldsymbol{w}}_i = \lfloor \boldsymbol{w}_i \rceil$ with the probability mass function (pmf) estimator $P_\theta(\tilde{\boldsymbol{w}}_i|\tilde{\boldsymbol{w}}_{<i}, \tilde{\boldsymbol{y}}_{<i})$ parameterized by θ, where $\lfloor\cdot\rceil$ is the rounding. Then, we obtain the residual latent $\boldsymbol{y}_i = f_\phi^y(\boldsymbol{x}, \boldsymbol{x}', \tilde{\boldsymbol{w}})$, where $f_\phi^y(\cdot)$ is the residual encoder. Then, similar to how we treat $\boldsymbol{w}_i$, we encode the quantized latent $\tilde{\boldsymbol{y}}_i = \lfloor \boldsymbol{y}_i \rceil$ with pmf $P_\theta(\tilde{\boldsymbol{y}}_i|\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{<i})$. Finally, we obtain the reconstructed frame $\boldsymbol{x}'_i = g_\theta^x(\boldsymbol{x}'_{i-1}, \tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{y}}_i)$, where $g_\theta^x(\cdot)$ is the decoder parameterized by θ.

As only the motion latent $\tilde{\boldsymbol{w}}_i$ and residual latent $\tilde{\boldsymbol{y}}_i$ exist in the bitstream, the above process can be simplified as Eq. 1 and Eq. 2, where $f_\phi(\cdot)$ is the generalized encoder and $g_\theta(\cdot)$ is the generalized decoder. The target of NVC is to minimize the per-frame R-D cost $R_i + \lambda_i D_i$ (Eq. 3), where R_i is the bitrate, D_i is the distortion and λ_i is the Lagrangian multiplier controlling R-D trade-off. The bitrate R_i and distortion D_i is computed as Eq. 2, where $d(\cdot,\cdot)$ is the distortion metric. And $\lambda_i D_i$ can be further interpreted as the data likelihood term $-\log p_\theta(\boldsymbol{x}_i|\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{\le i})$ so long as we treat $\lambda_i D_i$ as the energy function of a Gibbs distribution (Minnen et al., 2018). Specifically, when $d(\cdot,\cdot)$ is MSE, we can interpret $\lambda_i D_i = -\log p_\theta(\boldsymbol{x}_i|\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{\le i}) + const$, where $p_\theta(\boldsymbol{x}_i|\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{\le i})$ is a Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{x}}_i, 1/2\lambda_i I)$.

$$\boldsymbol{w}_i = f_\phi(\boldsymbol{x}_i, \tilde{\boldsymbol{w}}_{<i}, \tilde{\boldsymbol{y}}_{<i}), \boldsymbol{y}_i = f_\phi(\boldsymbol{x}_i, \tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{<i}), \text{where } \tilde{\boldsymbol{w}}_i = \lfloor \boldsymbol{w}_i \rceil, \tilde{\boldsymbol{y}}_i = \lfloor \boldsymbol{y}_i \rceil \tag{1}$$

$$R_i = \log P_\theta(\tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{y}}_i|\tilde{\boldsymbol{w}}_{<i}, \tilde{\boldsymbol{y}}_{<i}),\ D_i = d(\boldsymbol{x}_i, g_\theta(\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{\le i})) \tag{2}$$

$$\max -(R_i + \lambda_i D_i) \tag{3}$$

On the other hand, NVC is also closely related to Variational Autoencoder (VAE) (Kingma & Welling, 2013). As the rounding $\lfloor\cdot\rceil$ is not differentiable, Ballé et al. (2016); Theis et al. (2017) propose to relax it by additive uniform noise (AUN), and replace $\tilde{\boldsymbol{w}}_i = \lfloor \boldsymbol{w}_i \rceil$, $\tilde{\boldsymbol{y}}_i = \lfloor \boldsymbol{y}_i \rceil$ with $\tilde{\boldsymbol{w}}_i = \boldsymbol{w}_i + \mathcal{U}(-0.5, 0.5)$, $\tilde{\boldsymbol{y}}_i = \boldsymbol{y}_i + \mathcal{U}(-0.5, 0.5)$. Under such formulation, the above encoding-decoding process becomes a VAE on graphic model $\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{\le i} \to \boldsymbol{x}_i$ with variational posterior as Eq. 4, where $\boldsymbol{w}_i, \boldsymbol{y}_i$ plays the role of variational posterior parameter. Then, minimizing the overall R-D cost (Eq. 3) is equivalent to maximizing the evident lowerbound (ELBO) (Eq. 5).

$$q_\phi(\tilde{\boldsymbol{w}}_i|\boldsymbol{x}_i, \tilde{\boldsymbol{w}}_{<i}, \tilde{\boldsymbol{y}}_{<i}) = \mathcal{U}(\boldsymbol{w}_i - 0.5, \boldsymbol{w}_i + 0.5), q_\phi(\tilde{\boldsymbol{y}}_i|\boldsymbol{x}_i, \tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{<i}) = \mathcal{U}(\boldsymbol{y}_i - 0.5, \boldsymbol{y}_i + 0.5) \tag{4}$$

$$-(R_i + \lambda_i D_i) = \mathbb{E}_{q_\phi}[\underbrace{\log P_\theta(\tilde{\boldsymbol{w}}_i, \tilde{\boldsymbol{y}}_i|\tilde{\boldsymbol{w}}_{<i}, \tilde{\boldsymbol{y}}_{<i})}_{-R_i} + \underbrace{\log p_\theta(\boldsymbol{x}_i|\tilde{\boldsymbol{w}}_{\le i}, \tilde{\boldsymbol{y}}_{\le i})}_{-\lambda_i D_i} \underbrace{-\cancel{\log q_\phi}}_{\text{bits-back bitrate: 0}}] \tag{5}$$

2.2 BIT ALLOCATION FOR NEURAL VIDEO COMPRESSION

It is well known to video coding community that using the same R-D trade-off parameter λ_i to optimize R-D cost in Eq. 3 for all T frames inside a GoP is suboptimal (Li et al., 2014; 2016). This sub-optimality comes from the frame reference structure and is explained in detail by Li et al. (2022); Xu et al. (2022). The target of bit allocation is to maximize the minus of overall R-D cost

[1]Following previous works in deep generative modeling (Kingma & Welling, 2013; Kim et al., 2018), we denote all parameters related to encoder as ϕ, and all parameters related to decoder and prior as θ.

(ELBO) $\mathcal{L}$ as Eq. 6 given the overall R-D trade-off parameter λ_0, instead of maximizing $\mathcal{L}_i$ of each frame i separately.

The pioneer work of bit allocation in NVC (Li et al., 2022) follows bit allocation for traditional video codec (Li et al., 2016). Specifically, it adopts empirical models to approximate the relationship of the rate dependency $\partial R_{i+1}/\partial R_i$ and distortion dependency $\partial D_{i+1}/\partial D_i$ between frames. Then it takes those models into Eq. 6 to solve $\lambda^*_{1:T}$ explicitly as Eq. 7.*left*. However, its performance heavily relies on the accuracy of empirical models.

$$\max \mathcal{L} = \sum_{i=1}^{T} \mathcal{L}_i, \text{ where } \mathcal{L}_i = -(R_i + \lambda_0 D_i) \tag{6}$$

$$\lambda^*_{1:T} \leftarrow \arg\max_{\lambda_{1:T}} \mathcal{L}(\lambda_{1:T}), \text{ versus } \boldsymbol{w}^*_{1:T}, \boldsymbol{y}^*_{1:T} \leftarrow \arg\max_{\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}} \mathcal{L}(\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}) \tag{7}$$

On the other hand, BAO (Xu et al., 2022) does not solve $\lambda^*_{1:T}$ explicitly. Instead, it adopts SAVI (Kim et al., 2018; Marino et al., 2018) to achieve implicit bit allocation. To be specific, it initializes the variational posterior parameter $\boldsymbol{w}^0_{1:T}, \boldsymbol{y}^0_{1:T}$ from fully amortized variational inference (FAVI) as Eq. 1. Then, it optimizes $\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}$ via gradient ascent to maximize $\mathcal{L}$ as Eq. 7.*right*. During this procedure, no empirical model is required. BAO further proofs that optimizing Eq. 7.*right* is equivalent to optimizing Eq. 7.*left* with precise rate and distortion dependency model $\partial R_{i+1}/\partial R_i, \partial D_{i+1}/\partial D_i$ (See Thm. 1, Thm. 2 in Xu et al. (2022)). Thus, BAO claims that it is optimal assuming gradient ascent achieves global maximum. However, in next section, we show that BAO (Xu et al., 2022) is in fact suboptimal due to its implementation.

3 WHY BAO IS SUP-OPTIMAL

BAO (Xu et al., 2022) achieves the SAVI (Kim et al., 2018; Marino et al., 2018) target in Eq. 7.*right* by gradient-based optimization. More specifically, its update rule is described as Eq. 8 and Eq. 9, where K is the total number of gradient ascent steps, and $\boldsymbol{w}^k_i, \boldsymbol{y}^k_i$ is the posterior parameter $\boldsymbol{w}_i, \boldsymbol{y}_i$ after k steps of gradient ascent. In the original paper of BAO, the authors also find that directly optimizing $\boldsymbol{w}_i, \boldsymbol{y}_i$ simultaneously by Eq. 8 and Eq. 9 performs worse than optimizing $\boldsymbol{y}_i$ alone using Eq. 9, but they have not offered any explanation. It is obvious that optimizing $\boldsymbol{y}_i$ alone is sub-optimal. However, it is not obvious why jointly optimizing $\boldsymbol{w}_i, \boldsymbol{y}_i$ with Eq. 8 and Eq. 9 fails.

$$\boldsymbol{w}^{k+1}_i \leftarrow \boldsymbol{w}^k_i + \alpha \frac{d\mathcal{L}(\boldsymbol{w}^k_{1:T}, \boldsymbol{y}^k_{1:T})}{d\boldsymbol{w}^k_i}, \text{ where } \frac{d\mathcal{L}(\boldsymbol{w}^k_{1:T}, \boldsymbol{y}^k_{1:T})}{d\boldsymbol{w}^k_i} = \sum_{j=i}^{T} \frac{\partial \mathcal{L}_j(\boldsymbol{w}^k_{1:j}, \boldsymbol{y}^k_{1:j})}{\partial \boldsymbol{w}^k_i} \tag{8}$$

$$\boldsymbol{y}^{k+1}_i \leftarrow \boldsymbol{y}^k_i + \alpha \frac{d\mathcal{L}(\boldsymbol{w}^k_{1:T}, \boldsymbol{y}^k_{1:T})}{d\boldsymbol{y}^k_i}, \text{ where } \frac{d\mathcal{L}(\boldsymbol{w}^k_{1:T}, \boldsymbol{y}^k_{1:T})}{d\boldsymbol{y}^k_i} = \sum_{j=i}^{T} \frac{\partial \mathcal{L}_j(\boldsymbol{w}^k_{1:j}, \boldsymbol{y}^k_{1:j})}{\partial \boldsymbol{y}^k_i} \tag{9}$$

In fact, the update rule in Eq. 8 and Eq. 9 is exactly the SAVI (Kim et al., 2018; Marino et al., 2018) when $\boldsymbol{w}_i, \boldsymbol{y}_i$ fully factorizes (e.g. the full factorization used in mean-field (Blei et al., 2017)). However, in NVC the $\boldsymbol{w}_i, \boldsymbol{y}_i$ has complicated auto-regressive relationships (See Eq. 1 and Fig. 1.(a)). Abusing SAVI on non-factorized latent causes gradient error in two aspects: (1). The total derivative $d\mathcal{L}/d\boldsymbol{w}_i, d\mathcal{L}/d\boldsymbol{y}_i$ is incomplete. (2). The total derivative $d\mathcal{L}/d\boldsymbol{w}_i, d\mathcal{L}/d\boldsymbol{y}_i$ and partial derivative $\partial\mathcal{L}_j/\partial\boldsymbol{w}_i, \partial\mathcal{L}_j/\partial\boldsymbol{y}_i$ is evaluated at wrong value. In next two sections, we elaborate those two issues with $\boldsymbol{w}_i$ related equations in main text and $\boldsymbol{y}_i$ related equations in Appendix. A.2.

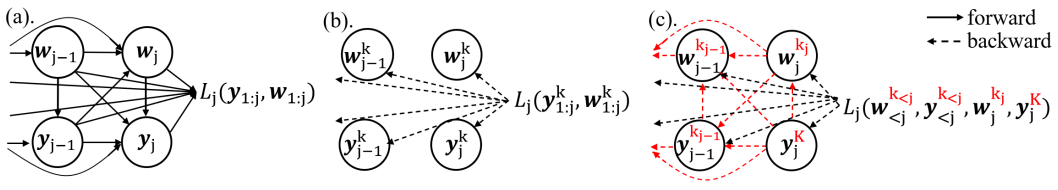

Figure 1: (a). The gradient structure of NVC without SAVI. (b). After k step of SAVI/gradient ascent, the gradient structure of NVC is broken. (c). The proposed approach using back-propagating through gradient ascent. We mark the difference between (b) and (c) in red.

3.1 INCOMPLETE TOTAL DERIVATIVE EVALUATION

According to the latent generation procedure described by Eq. 1 and Eq. 2, we draw the computational graph to describe the latent dependency as Fig. 1.(a). Based on that, we expand the total derivative $d\mathcal{L}/d\boldsymbol{w}_i, d\mathcal{L}/d\boldsymbol{y}_i$ as Eq. 10 and Eq. 22.

$$\frac{d\mathcal{L}(\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T})}{d\boldsymbol{w}_i} = \sum_{j=i}^{T} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{w}_i}$$

$$\frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{w}_i} = \underbrace{\sum_{l=i+1}^{j} \frac{\partial \boldsymbol{w}_l}{\partial \boldsymbol{w}_i} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{w}_l} + \sum_{l=i}^{j} \frac{\partial \boldsymbol{y}_l}{\partial \boldsymbol{w}_i} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{y}_l}}_{\text{ignored by BAO}} + \underbrace{\frac{\partial \mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{\partial \boldsymbol{w}_i}}_{\text{considered by BAO}} \tag{10}$$

As shown in Eq. 8, Eq. 9 and Fig. 1.(b), BAO (Xu et al., 2022) treats the total derivative $d\mathcal{L}/d\boldsymbol{w}_i, d\mathcal{L}/d\boldsymbol{y}_i$ as the sum of the frame level partial derivative $\partial\mathcal{L}_j/\partial\boldsymbol{w}_i, \partial\mathcal{L}_j/\partial\boldsymbol{y}_i$, which is the direct contribution of frame i^{th} latent $\boldsymbol{w}_i, \boldsymbol{y}_i$ to j^{th} frame's R-D cost $\mathcal{L}_j$ (as marked in Eq. 10 and Eq. 22). This incomplete evaluation of gradient signal brings sub-optimality. Further, it is not possible to correct BAO by simply including other parts of gradient into consideration. As BAO jointly updates all the latent $\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}$, the relationship of Eq. 2 only holds for the initial latent parameters $\boldsymbol{w}^0_{1:T}, \boldsymbol{y}^0_{1:T}$ produced by FAVI. And this important relationship is broken for parameters $\boldsymbol{w}^k_{1:T}, \boldsymbol{y}^k_{1:T}$ after $k \geq 1$ steps of update.

3.2 INCORRECT VALUE TO EVALUATE GRADIENT

As shown in Eq. 8 and Eq. 9, BAO (Xu et al., 2022) simultaneously updates all the posterior parameter $\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}$ with gradient evaluated at the same gradient ascent step $\boldsymbol{w}^k_{1:T}, \boldsymbol{y}^k_{1:T}$. However, as we show later in Sec. 4.1 and Fig. 1.(c), this is sub-optimal as all the descendant latent $\boldsymbol{w}_{>i}, \boldsymbol{y}_{\geq i}$ of $\boldsymbol{w}_i$ should already complete all K steps of gradient ascent before the gradient of $\boldsymbol{w}_i$ is evaluated. Moreover, $\boldsymbol{w}_{>i}, \boldsymbol{y}_{\geq i}$ should be initialized by FAVI using precedents latent. Similar rule applies to $\boldsymbol{y}_i$. Specifically, the correct value to evaluate the gradient is as Eq. 11 and Eq. 23, where $\boldsymbol{w}_i^{k_i}$ denotes the latent $\boldsymbol{w}_i$ after k_i steps of update, and $\boldsymbol{y}_i^{k'_j}$ denotes the latent $\boldsymbol{y}_i$ after k'_i steps of update.

$$\boldsymbol{w}_i^{k_i+1} \leftarrow \boldsymbol{w}_i^{k_i} + \alpha \frac{d\mathcal{L}(\boldsymbol{w}_1^{k_1}, ..., \boldsymbol{w}_i^{k_i}, \boldsymbol{w}_{>i}^{K}, \boldsymbol{y}_1^{k'_1}, ..., \boldsymbol{y}_{i-1}^{k'_{i-1}}, \boldsymbol{y}_{\geq i}^{K})}{d\boldsymbol{w}_i^{k_i}},$$

$$\text{where } \boldsymbol{w}^0_{>i}, \boldsymbol{y}^0_{\geq i} = f(\boldsymbol{x}, \boldsymbol{w}_1^{k_1}, ..., \boldsymbol{w}_i^{k_i}, \boldsymbol{y}_1^{k'_1}, ..., \boldsymbol{y}_{i-1}^{k'_{i-1}}) \tag{11}$$

Similar to the incomplete total derivative evaluation, this problem does not have a simple solution. In next section, we show how to correct both of the above-mentioned issues by recursively applying back-propagating through gradient ascent (Domke, 2012).

4 CORRECTING THE SUB-OPTIMAL BIT ALLOCATION

In this section, we first extend the generic SAVI Kim et al. (2018); Marino et al. (2018) to 2-level non-factorized latent. Then we further extend this result to latent with any dependency that can be described by a DAG (Directed Acyclic Graph). And finally, we correct the sub-optimal bit allocation by applying the result in DAG latent to NVC.

4.1 SAVI ON 2-LEVEL NON-FACTORIZED LATENT

In this section, we extend the SAVI on 1-level latent (Kim et al., 2018) to 2-level non-factorized latent. We denote $\boldsymbol{x}$ as evidence, $\boldsymbol{a}$ as the variational posterior parameter of the first level latent $\tilde{\boldsymbol{a}}$, $\boldsymbol{b}$ as the variational posterior parameter of the second level latent $\tilde{\boldsymbol{b}}$, and the ELBO to maximize as $\mathcal{L}(\boldsymbol{a}, \boldsymbol{b})$. The posterior $q(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}|\boldsymbol{x})$ factorizes as $q(\tilde{\boldsymbol{a}}|\boldsymbol{x})q(\tilde{\boldsymbol{b}}|\tilde{\boldsymbol{a}}, \boldsymbol{x})$, which means that $\boldsymbol{b}$ depends on $\boldsymbol{a}$. Given $\boldsymbol{a}$ is fixed, we can directly follow Kim et al. (2018); Marino et al. (2018) to optimize $\boldsymbol{b}$ to maximize ELBO by SAVI. However, it requires some tricks to optimize $\boldsymbol{a}$.

Algorithm 1: SAVI on 2-level Latent

procedure solve-2-level($\boldsymbol{x}, \boldsymbol{a}^k$)
  initialize $\boldsymbol{a}^0 \leftarrow f(\boldsymbol{x})$ from FAVI
  for $k = 0, ..., K-1$ **do**
    $\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{a}^k} = \text{grad-2-level}(\boldsymbol{x}, \boldsymbol{a}^k)$
    $\boldsymbol{a}^{k+1} \leftarrow \boldsymbol{a}^k + \alpha \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{a}^k}$
  return $\boldsymbol{a}^K, \boldsymbol{b}^K$

procedure grad-2-level($\boldsymbol{x}, \boldsymbol{a}^k$)
  $\boldsymbol{b}^0 \leftarrow f(\boldsymbol{x}, \boldsymbol{a}^k)$ from FAVI
  for $k' = 0, ..., K-1$ **do**
    $\boldsymbol{b}^{k'+1} \leftarrow \boldsymbol{b}^{k'} + \alpha \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{d\boldsymbol{b}^{k'}}$
  $\overleftarrow{\boldsymbol{a}} \leftarrow \frac{\partial \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{\partial \boldsymbol{a}^k}$
  $\overleftarrow{\boldsymbol{b}^K} \leftarrow \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{b}^K}$
  for $k' = K-1, ..., 0$ **do**
    $\overleftarrow{\boldsymbol{a}} \leftarrow \overleftarrow{\boldsymbol{a}} + \alpha \frac{\partial^2 \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{\partial \boldsymbol{a}^k \partial \boldsymbol{b}^{k'}} \overleftarrow{\boldsymbol{b}^{k'+1}}$
    $\overleftarrow{\boldsymbol{b}^{k'}} \leftarrow \overleftarrow{\boldsymbol{b}^{k'}} + \alpha \frac{\partial^2 \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{\partial \boldsymbol{b}^{k'} \partial \boldsymbol{b}^{k'}} \overleftarrow{\boldsymbol{b}^{k'+1}}$
  $\overleftarrow{\boldsymbol{a}} = \overleftarrow{\boldsymbol{a}} + \frac{\partial \boldsymbol{b}^0}{\partial \boldsymbol{a}^k} \overleftarrow{\boldsymbol{b}^0}$
  return $\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{a}^k} = \overleftarrow{\boldsymbol{a}}$

Algorithm 2: SAVI on DAG Latent

procedure solve-dag($\boldsymbol{x}$)
  sort $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ in topological order
  for $\boldsymbol{a}_j$ with parent $\mathcal{P}(\boldsymbol{a}_j) = \varnothing$
    add $\boldsymbol{a}_j$ to fake node $\boldsymbol{a}_0$'s children $\mathcal{C}(\boldsymbol{a}_0)$
  grad-dag($\boldsymbol{x}, \boldsymbol{a}_0^0$)
  return $\boldsymbol{a}_1^K, ..., \boldsymbol{a}_N^K$

procedure grad-dag($\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}$)
  for $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)$ in topological order **do**
    $\boldsymbol{a}_j^0 \leftarrow f(\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_{<j}^{k_{<j}})$ from FAVI
    for $k_j = 0, ..., K-1$ **do**
      $\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_j^{k_j}, \boldsymbol{a}_{>j}^K)}{d\boldsymbol{a}_j^{k_j}} \leftarrow \text{grad-dag}(\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_j^{k_j})$
      $\boldsymbol{a}_j^{k_j+1} \leftarrow \boldsymbol{a}_j^{k_j} + \alpha \frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_j^{k_j}, \boldsymbol{a}_{>j}^K)}{d\boldsymbol{a}_j^{k_j}}$
  $\overleftarrow{\boldsymbol{a}_i} \leftarrow \frac{\partial \mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{\partial \boldsymbol{a}_i^{k_i}}$
  for $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)$ **do**
    $\overleftarrow{\boldsymbol{a}_j} \leftarrow \boldsymbol{0}, \overleftarrow{\boldsymbol{a}_j^K} \leftarrow \frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}_j^K}$
    for $k_j = K-1, ..., 0$ **do**
      $\overleftarrow{\boldsymbol{a}_j} \leftarrow \overleftarrow{\boldsymbol{a}_j} + \alpha \frac{\partial^2 \mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_j^{k_j}, \boldsymbol{a}_{>j}^K)}{\partial \boldsymbol{a}_i^{k_i} \partial \boldsymbol{a}_j^{k_j}} \overleftarrow{\boldsymbol{a}_j^{k_j+1}}$
      $\overleftarrow{\boldsymbol{a}_j^{k_j}} \leftarrow \overleftarrow{\boldsymbol{a}_j^{k_j+1}} + \alpha \frac{\partial^2 \mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_j^{k_j}, \boldsymbol{a}_{>j}^K)}{\partial \boldsymbol{a}_j^{k_j} \partial \boldsymbol{a}_j^{k_j}} \overleftarrow{\boldsymbol{a}_j^{k_j+1}}$
    $\overleftarrow{\boldsymbol{a}_i} \leftarrow \overleftarrow{\boldsymbol{a}_i} + \overleftarrow{\boldsymbol{a}_j} + \frac{\partial \boldsymbol{a}_j^0}{\partial \boldsymbol{a}_i^{k_i}} \overleftarrow{\boldsymbol{a}_j^0}$
  return $\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}_i^{k_i}} = \overleftarrow{\boldsymbol{a}_i}$

The intuition is, we do not want to find a $\boldsymbol{a}$ that maximizes $\mathcal{L}(\boldsymbol{a}, \boldsymbol{b})$ given a fixed $\boldsymbol{b}$ (or we have the gradient issue described in Sec. 3). Instead, we want to find a $\boldsymbol{a}$, whose $\max_{\boldsymbol{b}} \mathcal{L}(\boldsymbol{a}, \boldsymbol{b})$ is maximum. This translates to the optimization problem as Eq. 12. In fact, Eq. 12 is a variant of setup in back-propagating through gradient ascent (Samuel & Tappen, 2009; Domke, 2012). The difference is, our $\boldsymbol{a}$ also contributes directly to optimization target $\mathcal{L}(\boldsymbol{a}, \boldsymbol{b})$. From this perspective, Eq. 12 is more closely connected to Kim et al. (2018), if we treat $\boldsymbol{a}$ as the model parameter and $\boldsymbol{b}$ as latent.

$$\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}} \mathcal{L}(\boldsymbol{a}, \boldsymbol{b}^*(\boldsymbol{a})), \text{where } \boldsymbol{b}^*(\boldsymbol{a}) \leftarrow \arg\max_{\boldsymbol{b}} \mathcal{L}(\boldsymbol{a}, \boldsymbol{b}) \tag{12}$$

And as SAVI on 1-level latent (Kim et al., 2018; Marino et al., 2018), we need to solve Eq. 12 using gradient ascent. Specifically, denote α as step size (learning rate), K as the total gradient ascent steps, $\boldsymbol{a}^k$ as the $\boldsymbol{a}$ after k step update, $\boldsymbol{b}^{k'}$ as the $\boldsymbol{b}$ after k' step update, and $f(.)$ as FAVI procedure generating initial posterior parameters $\boldsymbol{a}^0, \boldsymbol{b}^0$, the optimization problem as Eq. 12 translates into the update rule as Eq. 13. Eq. 13 is the guidance for designing optimization algorithm, and it also explains why the gradient of BAO (Xu et al., 2022) is evaluated at wrong value (See Sec. 3.2).

$$\boldsymbol{a}^{k+1} \leftarrow \boldsymbol{a}^k + \alpha \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{a}^k}, \boldsymbol{b}^{k'+1} \leftarrow \boldsymbol{b}^{k'} + \alpha \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{d\boldsymbol{b}^{k'}}, \text{where } \boldsymbol{b}^0 = f(\boldsymbol{x}, \boldsymbol{a}^k) \tag{13}$$

To solve Eq. 13, we note that although $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})/d\boldsymbol{b}^{k'}$ is directly computed, $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{a}^k$ is not straightforward. Resorting to previous works (Samuel & Tappen, 2009; Domke, 2012) in implicit differentiation and extending the results in Kim et al. (2018) from model parameters to variational posterior parameters, we implement Eq. 13 as Alg. 1. Specifically, we first initialize $\boldsymbol{a}^0$ from FAVI. Then we conduct gradient ascent on $\boldsymbol{a}$ with gradient $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{a}^k$ computed from the procedure grad-2-level($\boldsymbol{x}, \boldsymbol{a}^k$). And inside grad-2-level($\boldsymbol{x}, \boldsymbol{a}^k$), $\boldsymbol{b}$ is also updated by gradient

ascent, the above procedure corresponds to Eq. 13. The key of Alg. 1 is the evaluation of gradient $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{a}^k$. Formally, we have:

Theorem 1. *After* grad-2-level$(\boldsymbol{x}, \boldsymbol{a}^k)$ *of Alg. 1 executes, we have the return value* $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{a}^k = \overleftarrow{\boldsymbol{a}}$. *(See proof in Appendix. A.1.)*

4.2 SAVI ON DAG-DEFINED NON-FACTORIZED LATENT

In this section, we extend the result from previous section to SAVI on general non-factorized latent with dependency described by any 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 is the general case covering all dependency that can be described by DGM. This extension is necessary to perform 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 $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ with their dependency defined by a computational graph $\mathcal{G}$, i.e., their corresponding latent variable $\tilde{\boldsymbol{a}}_1, ..., \tilde{\boldsymbol{a}}_N$'s posterior distribution factorizes as $\mathcal{G}$. Specifically, we denote $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i), \boldsymbol{a}_i \in \mathcal{P}(\boldsymbol{a}_j)$ if an edge exists from $\boldsymbol{a}_i$ to $\boldsymbol{a}_j$. This indicates that $\tilde{\boldsymbol{a}}_j$ conditions on $\tilde{\boldsymbol{a}}_i$. Without loss of generality, we assume $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ is sorted in topological order. This means that if $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i), \boldsymbol{a}_i \in \mathcal{P}(\boldsymbol{a}_j)$, then $i < j$. Each latent is optimized by K-step gradient ascent, and $\boldsymbol{a}_i^{k_i}$ denotes the latent $\boldsymbol{a}_i$ after k_i steps of update. Then, similar to 2-level latent, we have the update rule as Eq. 14:

$$\boldsymbol{a}_i^{k_i+1} \leftarrow \boldsymbol{a}_i^{k_i} + \alpha \frac{d\mathcal{L}(\boldsymbol{a}_1^{k_1}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}^{k_i}}, \text{ where } \boldsymbol{a}_{>i}^0 = f(\boldsymbol{x}, \boldsymbol{a}_1^{k_1}, ..., \boldsymbol{a}_i^{k_i}) \tag{14}$$

, which can be translated into Alg. 2. Specifically, we first sort the latent in topological order. Then, we add a fake latent $\boldsymbol{a}_0$ to the front of all $\boldsymbol{a}$s. Its children are all the $\boldsymbol{a}s$ with 0 in-degree. Then, we can solve the SAVI on $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ using gradient ascent by executing the procedure grad-dag$(\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i})$ in Alg. 2 recursively. Inside procedure grad-dag$(\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i})$, the gradient to update $\boldsymbol{a}_i$ relies on the convergence of its children $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)$, which is implemented by the recursive depth-first search (DFS) in line 11. And upon the completion of procedure grad-dag$(\boldsymbol{x}, \boldsymbol{a}_0^0)$, all the latent converges to $\boldsymbol{a}_1^K, ..., \boldsymbol{a}_N^K$. Similar to the 2-level latent case, the key of Alg. 2 is the evaluation of gradient $d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)/d\boldsymbol{a}_i^{k_i}$. Formally, we have:

Theorem 2. *After the procedure* grad-dag$(\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i})$ *in Alg. 2 executes, we have the return value* $d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)/d\boldsymbol{a}_i^{k_i} = \overleftarrow{\boldsymbol{a}_i}$. *(See proof in Appendix. A.1.)*

To better understand how Alg. 2 works, we provide a detailed example in Fig. 5 of Appendix. A.3.

4.3 CORRECTING THE SUB-OPTIMAL BIT ALLOCATION USING SAVI ON DAG

With the result in previous section, correcting BAO (Xu et al., 2022) seems to be trivial. We only need to sort the latent in topological order as $\boldsymbol{w}_1, \boldsymbol{y}_1, ..., \boldsymbol{w}_T, \boldsymbol{y}_T$, treat them as $\boldsymbol{a}_1, ..., \boldsymbol{a}_{2T+1}$ and run Alg. 2 to obtain the optimized latent parameters $\boldsymbol{w}_1^K, \boldsymbol{y}_1^K, ..., \boldsymbol{w}_T^K, \boldsymbol{y}_T^K$. And the gradient $d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)/d\boldsymbol{a}_i^{k_i}$ computed in Alg. 2 resolves the issue of BAO described in Sec. 3.1 and Sec. 3.2. However, an evident problem is the temporal complexity. Given the latent number N and gradient ascent step number K, Alg. 2 has temporal complexity of $\Theta(K^N)$. NVC with GoP size 10 has approximately $N = 20$ latent, and the SAVI on NVC (Xu et al., 2022) takes around $K = 2000$ step to converge. For bit allocation, the complexity of Alg. 2 is $\approx 2000^{20}$, which is intractable. On the other hand, BAO's complexity is reasonable ($\Theta(KN) \approx 4 \times 10^4$). Thus, in next section, we provide a feasible approximation to such intractable corrected bit allocation.

4.4 FEASIBLE APPROXIMATION TO THE CORRECTED BIT ALLOCATION

In order to solve problem with practical size such as bit allocation on NVC, we provide an approximation to the SAVI (Kim et al., 2018; Marino et al., 2018) on DAG described in Sec. 4.2. The general idea is that, when being applied to bit allocation of NVC, the accurate SAVI on DAG (Alg. 2) satisfies both requirement on gradient signal described in Sec. 3.1 and Sec. 3.2. We can not make it

tractable without breaking them. Thus, we break one of them and achieve a reasonable complexity, while maintain a superior performance compared with BAO (Xu et al., 2022).

We consider the approximation in Eq. 15 which breaks the requirement for gradient evaluation in Sec. 3.2. Based on Eq. 15 and the requirement in Sec. 3.1, we design an approximation of accurate SAVI as Alg. 4. When being applied to bit allocation in NVC, it satisfies the gradient requirement in Sec. 3.1 while maintaining a temporal complexity of $\Theta(KN)$ as BAO.

$$\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}_i^{k_i}} \approx \frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^0)}{d\boldsymbol{a}_i^{k_i}} \tag{15}$$

Specifically, with the approximation in Eq. 15, the recurrent gradient computation in Alg. 2 becomes unnecessary as the right hand side of Eq. 15 does not require $\boldsymbol{a}_{>i}^K$. However, to maintain the dependency of latent described in Sec. 3.1, as Alg. 2, we still need to ensure that the children node $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)$ are re-initialized by FAVI every-time when $\boldsymbol{a}_i$ is updated. Therefore, a reasonable approach is to traverse the graph in topological order. We keep the children node $\boldsymbol{a}_j$ untouched until all its parent node $\boldsymbol{a}_i \in \mathcal{P}(\boldsymbol{a}_j)$'s gradient ascent is completed and $\boldsymbol{a}_i^K$ is known. And the resulting approximate SAVI algorithm is as Alg. 4. When applied to bit allocation, it satisfies the gradient requirement in Sec. 3.1, and as BAO, its temporal complexity is $\Theta(KN)$.

Algorithm 3: BAO on DAG Latent

procedure solve-bao($\boldsymbol{x}$)
  $\boldsymbol{a}_1^0, ..., \boldsymbol{a}_N^0 \leftarrow f(\boldsymbol{x})$ from FAVI
  for $k = 0, ..., K-1$ **do**
    for $i = 1, ..., N$ **do**
      $\boldsymbol{a}_i^{k+1} \leftarrow \boldsymbol{a}_i^k + \alpha \frac{\partial \mathcal{L}(\boldsymbol{a}_1^k, ..., \boldsymbol{a}_N^k)}{\partial \boldsymbol{a}_i^k}$
  return $a_1^K, ..., a_N^K$

Algorithm 4: Approximate SAVI on DAG latent

procedure solve-approx-dag($\boldsymbol{x}$)
  sort $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ in topological order
  for $i = 1, ..., N$ **do**
    $\boldsymbol{a}_i^0, ..., \boldsymbol{a}_N^0 \leftarrow f(\boldsymbol{x}, \boldsymbol{a}_{<i}^K)$ from FAVI
    for $k = 0, ..., K-1$ **do**
      $\frac{d\mathcal{L}(\boldsymbol{a}_{<i}^K, \boldsymbol{a}_i^k, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}_i^k} \approx \frac{d\mathcal{L}(\boldsymbol{a}_{<i}^K, \boldsymbol{a}_i^k, \boldsymbol{a}_{>i}^0)}{d\boldsymbol{a}_i^k}$
      $\boldsymbol{a}_i^{k+1} \leftarrow \boldsymbol{a}_i^k + \alpha \frac{d\mathcal{L}(\boldsymbol{a}_{<i}^K, \boldsymbol{a}_i^k, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}_i^k}$
  return $a_1^K, ..., a_N^K$

To better understand BAO (Xu et al., 2022) in SAVI context, we rewrite it by general SAVI notation instead of NVC notation in Alg. 3. We highlight the difference between BAO (Alg. 3) (Xu et al., 2022), the accurate SAVI on DAG latent (Alg. 2) and the approximate SAVI on DAG latent (Alg. 4) from several aspects:

- **Graph Traversal Order**: BAO performs gradient ascent on $\boldsymbol{a}_{1:T}$ all together. The accurate SAVI only updates $\boldsymbol{a}_i$ when $\boldsymbol{a}_{>i}$'s update is complete and $\boldsymbol{a}_{>i}^K$ is known. The approximate SAVI only updates $\boldsymbol{a}_i$ when $\boldsymbol{a}_{<i}$'s update is complete and $\boldsymbol{a}_{<i}^K$ is known.
- **Gradient Correctness**: When being applied to bit allocation in NVC, BAO violates the gradient rule in Sec. 3.1 and Sec. 3.2, accurate SAVI satisfies both rules, approximate SAVI satisfies Sec. 3.1 and violates Sec. 3.2.
- **Temporal Complexity**: With the latent number N and steps of gradient ascent K, the complexity of BAO is $\Theta(KN)$, the complexity of accurate SAVI is $\Theta(K^N)$ and the complexity of approximate SAVI is $\Theta(KN)$.

Then we can simply apply Alg. 4 to bit allocation in NVC to obtain a feasible approximation of the corrected optimal bit allocation. And in Sec. 6.2, we empirically show that our approximation improves the R-D performance over BAO (Xu et al., 2022) with even smaller number of updates.

5 Related Work: Bit Allocation & SAVI for Neural Compression

Li et al. (2022) are the pioneer of bit allocation for NVC and their work is elaborated in Sec. 2.2. Other recent works that consider bit allocation for NVC only adopt simple heuristic such as inserting 1 high quality frame per 4 frames (Hu et al., 2022; Cetin et al., 2022). On the other hand, OEU (Lu et al., 2020) is also recognised as frame-level bit allocation while its performance is inferior than BAO (Xu et al., 2022). BAO is the most recent work with best R-D performance. It is elaborated in Sec. 2.2 and Sec. 3, and corrected in the previous section.

Semi-Amortized Variational Inference (SAVI) is proposed by Kim et al. (2018); Marino et al. (2018). The idea is that works following Kingma & Welling (2013) use fully amortized inference parameter ϕ 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 model parameters. We adopt a similar method to extend SAVI to non-factorized latent. When applying SAVI to practical neural codec, researchers abandon the nested model 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. And BAO (Xu et al., 2022) is the first to consider SAVI for bit allocation.

6 Experiments

6.1 Experimental Settings

We implement our approach in PyTorch 1.9 with CUDA 11.2, and run the experiments on NVIDIA(R) A100 GPU. Most of the other settings are intentionally kept the same as BAO (Xu et al., 2022). Specifically, we adopt HEVC Common Testing Condition (CTC) (Bossen et al., 2013) and UVG dataset (Mercat et al., 2020). And we measure the R-D performance in Bjontegaard-Bitrate (BD-BR) and BD-PSNR (Bjontegaard, 2001). For baseline NVC (Lu et al., 2019; Li et al., 2021), we adopt the official pre-trained models. And we select target $\lambda_0 = \{256, 512, 1024, 2048\}$. For gradient ascent, we adopt Adam (Kingma & Ba, 2014) optimizer with $lr = 1 \times 10^{-3}$. We set the gradient ascent step $K = 2000$ for the first frame and $K = 400$ for other frames. More details are presented in Appendix. A.5.

6.2 Quantitative Results

As shown in Tab. 1, our method consistently improves the R-D performance in terms of BD-BR over BAO (Xu et al., 2022) on both baseline methods and all datasets. Moreover, this improvement is especially significant (more than 10% in BD-BR) when the baseline is DCVC (Li et al., 2021). And both BAO and our proposed correction significantly outperform other approaches. It is also noteworthy that with our bit allocation, DVC (the SOTA method in 2019) already outperforms DCVC (the SOTA method in 2021) by large margin (See the red solid line and black dash line in Fig. 2).

Method	BD-BR (%) ↓ Class B	Class C	Class D	Class E	UVG
DVC (Lu et al., 2019) as Baseline					
Li et al. (2016)[1]	20.21	17.13	13.71	10.32	16.69
Li et al. (2022)[1]	-6.80	-2.96	0.48	-6.85	-4.12
OEU (Lu et al., 2020)[2]	-13.57	-11.29	-18.97	-12.43	-13.78
BAO (Xu et al., 2022)[2]	-28.55	-26.82	-25.37	-32.54	-27.68
Proposed	-32.10	-31.71	-35.86	-32.93	-30.92
DCVC (Li et al., 2021) as Baseline					
OEU (Lu et al., 2020)[2]	-10.75	-14.34	-16.30	-7.15	-16.07
BAO (Xu et al., 2022)[2]	-20.59	-19.69	-20.60	-23.33	-25.22
Proposed	-32.89	-33.10	-32.01	-36.88	-39.66

Table 1: The BD-BR of our approach compared with others. [1] comes from Li et al. (2022). [2] comes from Xu et al. (2022).

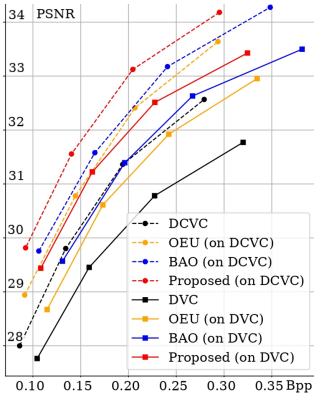

Figure 2: The R-D curve on HEVC Class D.

Other than R-D performance, the bitrate error of our approach is also significantly smaller than BAO (Xu et al., 2022) (See Tab. 2). The bitrate error is measured as the relative bitrate difference before and after bit allocation. The smaller it is, the easier it is to achieve the desired bitrate accurately. For complexity, our approach only performs 920 steps of gradient ascent per-frame, while BAO requires 2000 steps.

See more quantitative results (BD-PSNR & R-D curves) in Appendix. A.6.

6.3 Ablation Study, Analysis & Qualitative Results

Tab. 3 shows that for BAO (Xu et al., 2022), jointly optimizing $\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}$ performs worse than optimizing $\boldsymbol{y}_{1:T}$ or $\boldsymbol{w}_{1:T}$ alone. This counter-intuitive phenomena comes from its incorrect estimation of gradient signal. For the proposed approach that corrects this, jointly optimizing $\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T}$ performs better than optimizing $\boldsymbol{y}_{1:T}$ or $\boldsymbol{w}_{1:T}$ alone, which is aligned with our intuition.

	Bitrate-Error (%) ↓				
Method	Class B	Class C	Class D	Class E	UVG
DVC (Lu et al., 2019) as Baseline					
BAO (Xu et al., 2022)[2]	8.41	12.86	21.39	5.94	3.73
Proposed	3.16	4.27	1.81	6.14	1.73
DCVC (Li et al., 2021) as Baseline					
BAO (Xu et al., 2022)[2]	25.67	23.90	23.74	24.88	21.86
Proposed	4.27	7.29	5.73	8.03	3.06

Table 2: The bitrate error of our approach compared with BAO.

Method	BD-BR (%) ↓
BAO ($\boldsymbol{y}$)	-25.37
BAO ($\boldsymbol{w}$)	-22.24
BAO ($\boldsymbol{y}, \boldsymbol{w}$)	-14.76
Proposed ($\boldsymbol{y}$)	-32.60
Proposed ($\boldsymbol{w}$)	-31.56
Proposed ($\boldsymbol{y}, \boldsymbol{w}$)	-35.86

Table 3: Ablation study with HEVC Class D and DVC (Lu et al., 2019).

To better understand why our method works, we present the R-D cost, distortion and rate versus frame/latent index for different methods in Fig. 3: *top-left* shows that the R-D cost of our approach consistently decreases according to SAVI stage. Moreover, it outperforms BAO after 4^{th} frame; *top-right* shows that for each frame the R-D cost of our method is lower than BAO; *bottom-left* shows that the distortion part of R-D cost of our approach is approximately the same as BAO. While *bottom-right* shows that the advantage of our approach over BAO lies in the bitrate. More specifically, BAO increases the bitrate of $\boldsymbol{y}_i$s after SAVI, while our correction decreases it.

See more analysis in Appendix. A.9 and qualitative results in Appendix. A.10.

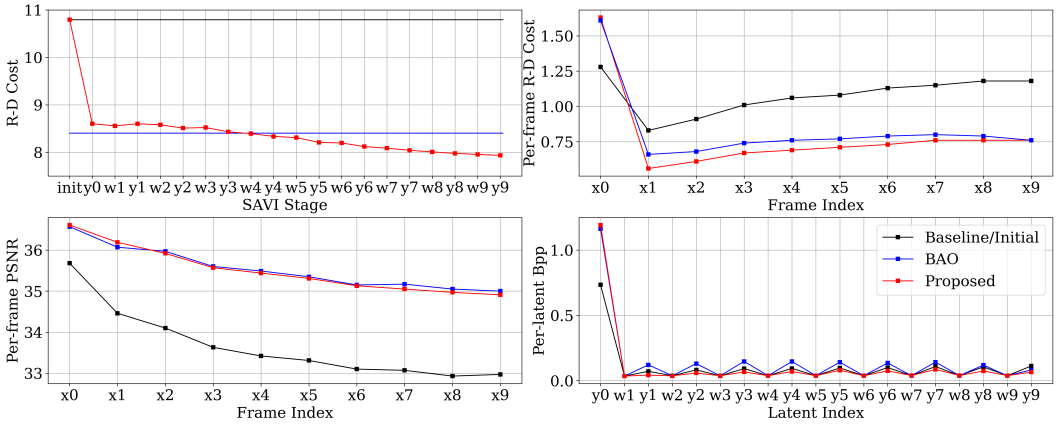

Figure 3: *top-left*. R-D cost vs. SAVI stage. *top-right*. R-D cost vs. frame index. *bottom-left*. PSNR vs. frame index. *bottom-right*. bpp vs. latent index. See enlarged-version in Appendix. A.9.

7 Discussion & Conclusion

Despite our correction is already more efficient than original BAO (Xu et al., 2022), its encoding speed remains far from real-time. Thus, it is limited to scenarios where R-D performance matters much more than encoding time (e.g. video on demand). See more discussion in Appendix. A.11.

To conclude, we show that a previous bit allocation method for NVC is sub-optimal as it abuses SAVI on non-factorized latent. Then, we propose the correct SAVI on general non-factorized latent by back-propagating through gradient ascent, and we further propose a feasible approximation to make it tractable for bit allocation. Experimental results show that our correction significantly improves the R-D performance.

ETHICS STATEMENT

Improving the R-D performance of NVC has positive social value, in terms of reducing carbon emission by saving the resources required to transfer and store videos. Moreover, unlike traditional codecs such as H.266 (Bross et al., 2021), neural video codec does not require dedicated hardware. Instead, it can be deployed with general neural accelerators. Improving the R-D performance of NVC prompts the practical deployment of video codecs that are independent of dedicated hardware, and lowers the hardware-barrier of playing multi-media contents.

REPRODUCIBILITY STATEMENT

For theoretical results, both of the two theorems are followed by proof in Appendix. A.1. For a relatively complicated novel algorithm (Alg. 2), we provide an illustration of the step by step execution procedure in Appendix. A.3. For experiment, both of the two datasets are publicly accessible. In Appendix. A.5, we provide more implementation details including all the hyper-parameters. Moreover, we provide our source code for reproducing the empirical results in supplementary material.

REFERENCES

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Frank Bossen et al. Common test conditions and software reference configurations. *JCTVC-L1100*, 12(7), 2013.

Benjamin Bross, Jianle Chen, Jens-Rainer Ohm, Gary J Sullivan, and Ye-Kui Wang. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). *Proc. IEEE*, 109(9):1463–1493, 2021.

Eren Cetin, M Akın Yılmaz, and A Murat Tekalp. Flexible-rate learned hierarchical bi-directional video compression with motion refinement and frame-level bit allocation. *arXiv e-prints*, pp. arXiv–2206, 2022.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *Int. Conf. on Machine Learning*, pp. 1078–1086. PMLR, 2018.

Joaquim Campos Meierhans Simon Abdelaziz Djelouah and Christopher Schroers. Content adaptive optimization for neural image compression. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, 2019.

Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pp. 318–326. PMLR, 2012.

Chenjian Gao, Tongda Xu, Dailan He, Hongwei Qin, and Yan Wang. Flexible neural image compression via code editing. *arXiv preprint arXiv:2209.09422*, 2022.

Zhihao Hu, Guo Lu, Jinyang Guo, Shan Liu, Wei Jiang, and Dong Xu. Coarse-to-fine deep video coding with hyperprior-guided mode prediction. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, pp. 5921–5930, 2022.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *Int. Conf. on Machine Learning*, pp. 2678–2687. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Bin Li, Houqiang Li, Li Li, and Jinlei Zhang. λ domain rate control algorithm for high efficiency video coding. *IEEE Trans. Image Process.*, 23(9):3841–3854, 2014.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34, 2021.

Li Li, Bin Li, Houqiang Li, and Chang Wen Chen. λ-domain optimal bit allocation algorithm for high efficiency video coding. *IEEE Trans. Circuits Syst. Video Technol.*, 28(1):130–142, 2016.

Yanghao Li, Xinyao Chen, Jisheng Li, Jiangtao Wen, Yuxing Han, Shan Liu, and Xiaozhong Xu. Rate control for learned video compression. In *ICASSP 2022-2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2829–2833. IEEE, 2022.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, pp. 11006–11015, 2019.

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 *European Conference on Computer Vision*, pp. 456–472. Springer, 2020.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *Int. Conf. on Machine Learning*, pp. 3403–3412. PMLR, 2018.

Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 297–302, 2020.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

Kegan GG Samuel and Marshall F Tappen. Learning optimized map estimates in continuously-valued mrf models. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 477–484. IEEE, 2009.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *5th Int. Conf. on Learning Representations, ICLR 2017*, 2017.

Tongda Xu, Han Gao, Chenjian Gao, Jinyong Pi, Yanghao Li, Yuanyuan Wang, Ziyu Zhu, Dailan He, Mao Ye, Hongwei Qin, and Yan Wang. Bit allocation using optimization. *arXiv preprint arXiv:2209.09422*, 2022.

Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt. Hierarchical autoregressive modeling for neural video compression. *arXiv preprint arXiv:2010.10258*, 2020a.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. *Advances in Neural Information Processing Systems*, 33:573–584, 2020b.

Jing Zhao, Bin Li, Jiahao Li, Ruiqin Xiong, and Yan Lu. A universal encoder rate distortion optimization framework for learned compression. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit*, pp. 1880–1884, 2021.

A APPENDIX

A.1 PROOF OF THM 1 AND THM 2

Theorem 1. *After the procedure* grad-2-level($\boldsymbol{x}, \boldsymbol{a}^k$) *of Alg. 1 executes, we have the return value* $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{a}^k = \overleftarrow{\boldsymbol{a}}$.

Proof. This proof extends the proof of Thm. 1 in Domke (2012), and it also serves as a formal justification of Alg. 1 in Kim et al. (2018). Note that our paper and Kim et al. (2018) are subtly different from Samuel & Tappen (2009); Domke (2012) as our high level parameter $\boldsymbol{w}$ not only generate low level parameter $\boldsymbol{y}$, but also directly contributes to optimization target (See Fig. 4).

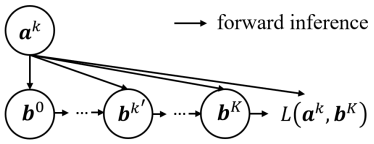

Figure 4: The computational graph corresponding to Eq. 16

As the computational graph in Fig. 4 shows, we can expand $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{a}^k$ as Eq. 16, with each term solved in Eq. 18 and Eq. 19.

$$\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{a}^k} = \underbrace{\frac{\partial \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{\partial \boldsymbol{a}^k}}_{\text{known}} + \sum_{k'=0}^{K} \underbrace{\frac{\partial \boldsymbol{b}^{k'}}{\partial \boldsymbol{a}^k}}_{\text{Eq. 18}} \underbrace{\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{b}^{k'}}}_{\text{Eq. 19}} \tag{16}$$

To solve Eq. 16, we first note that $\partial \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/\partial \boldsymbol{a}^k, d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{b}^K, \partial \boldsymbol{b}^0/\partial \boldsymbol{a}^k$ is naturally known. Then, by taking partial derivative of the update rule of gradient ascent $\boldsymbol{b}^{k'+1} \leftarrow \boldsymbol{b}^{k'} + \alpha d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})/d\boldsymbol{b}^{k'}$ with regard to $\boldsymbol{a}^k, \boldsymbol{b}^{k'}$, we have Eq. 17 and Eq. 18. Note that Eq. 18 is the partial derivative $\partial \boldsymbol{b}^{k'+1}/\partial \boldsymbol{a}^k$ instead of total derivative $d\boldsymbol{b}^{k'+1}/d\boldsymbol{a}^k = (\partial \boldsymbol{b}^{k'+1}/\partial \boldsymbol{b}^{k'})(d\boldsymbol{b}^{k'}/d\boldsymbol{a}^k) + \partial \boldsymbol{b}^{k'+1}/\partial \boldsymbol{a}^k$.

$$\frac{\partial \boldsymbol{b}^{k'+1}}{\partial \boldsymbol{b}^{k'}} = I + \alpha \frac{\partial^2 \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{\partial \boldsymbol{b}^{k'} \partial \boldsymbol{b}^{k'}} \tag{17}$$

$$\frac{\partial \boldsymbol{b}^{k'+1}}{\partial \boldsymbol{a}^k} = \alpha \frac{\partial^2 \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{\partial \boldsymbol{a}^k \partial \boldsymbol{b}^{k'}} \tag{18}$$

And those second order terms can either be directly evaluated or approximated via finite difference as Eq. 20. As Eq. 18 already solves the first term on the right hand side of Eq. 16, the remaining issue is $d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{b}^{k'}$. To solve this term, we expand it recursively as Eq. 19 and take Eq. 17 into it.

$$\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{b}^{k'}} = \frac{\partial \boldsymbol{b}^{k'+1}}{\partial \boldsymbol{b}^{k'}} \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)}{d\boldsymbol{b}^{k'+1}} \tag{19}$$

And the above solving process can be described by the procedure grad-2-level($\boldsymbol{x}, \boldsymbol{a}^k$) of Alg. 1. Specifically, the iterative update of $\overleftarrow{\boldsymbol{b}^{k'+1}}$ in line 15 corresponds to recursively expanding Eq. 19 with Eq. 17, and the iterative update of $\overleftarrow{\boldsymbol{a}}$ in line 14 corresponds to recursively expanding Eq. 16 with Eq. 18 and Eq. 19. Upon the return of grad-2-level($\boldsymbol{x}, \boldsymbol{a}^k$) of Alg. 1, we have $\overleftarrow{\boldsymbol{a}} = d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^K)/d\boldsymbol{b}^k$.

The complexity of the Hessian-vector product in line 14 and 15 of Alg. 1 may be reduced using finite difference following (Domke, 2012) as Eq. 20.

$$\begin{aligned} \frac{\partial^2 \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{\partial \boldsymbol{a}^k \partial \boldsymbol{b}^{k'}} \boldsymbol{v} &= \lim_{r \to 0} \frac{1}{r} \left(\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'} + r\boldsymbol{v})}{d\boldsymbol{a}^k} - \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{d\boldsymbol{a}^k} \right) \\ \frac{\partial^2 \mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{\partial \boldsymbol{b}^{k'} \partial \boldsymbol{b}^{k'}} \boldsymbol{v} &= \lim_{r \to 0} \frac{1}{r} \left(\frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'} + r\boldsymbol{v})}{d\boldsymbol{b}^{k'}} - \frac{d\mathcal{L}(\boldsymbol{a}^k, \boldsymbol{b}^{k'})}{d\boldsymbol{b}^{k'}} \right) \end{aligned} \tag{20}$$

□

Theorem 2. *After the procedure* grad-dag$(\boldsymbol{x}, \boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i})$ *in Alg. 2 executes, we have the return value* $d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)/d\boldsymbol{a}_i^{k_i} = \overleftarrow{\boldsymbol{a}}_i$.

Proof. Consider computing the target gradient with DAG $\mathcal{G}$. The $\boldsymbol{a}_i^k$'s gradient is composed of its own contribution to $\mathcal{L}$ in addition to the gradient from its children $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)$. Further, as we are considering the optimized children $\boldsymbol{a}_j^K$, we expand the children node $\boldsymbol{a}_j$ as Fig. 4. Then, we have:

$$\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{d\boldsymbol{a}_i^{k_i}} = \underbrace{\frac{\partial \mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_i^{k_i}, \boldsymbol{a}_{>i}^K)}{\partial \boldsymbol{a}_i^{k_i}}}_{\text{known}} + \sum_{\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)} \Big(\sum_{k_j=0}^{K} \underbrace{\frac{\partial \boldsymbol{a}_j^{k_j}}{\partial \boldsymbol{a}_i^{k_i}}}_{\text{Eq. 18}} \underbrace{\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_{j-1}^{k_{j-1}}, \boldsymbol{a}_{\geq j}^K)}{d\boldsymbol{a}_j^{k_j}}}_{\text{Eq. 19}} \Big) \tag{21}$$

The first term on the right-hand side of Eq. 21 can be trivially evaluated. The $\partial \boldsymbol{a}_j^{k_j}/\partial \boldsymbol{a}_i^{k_i}$ can be evaluated as Eq. 18. And the $d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_{j-1}^{k_{j-1}}, \boldsymbol{a}_{\geq j}^K)/d\boldsymbol{a}_j^{k_j}$ can be iteratively expanded as Eq. 19. We highlight several key differences between Alg. 2 and Alg. 1 which are reflected in the implementation of Alg. 2:

- The gradient evaluation of current node $\boldsymbol{y}_i$ requires gradient of its plural direct children $\boldsymbol{a}_j \in \mathcal{C}(\boldsymbol{a}_i)$, instead of the single child in 2-level case. The children traversal part of Eq. 19 corresponds to the two extra for loop in line 8 and 14 of Alg. 2.
- The gradient ascent update of child latent parameter $\boldsymbol{a}_j^{k_j+1} \leftarrow \boldsymbol{a}_j^{k_j} + \alpha d\mathcal{L}(\boldsymbol{a}_0^{k_0}, ..., \boldsymbol{a}_j^{k_j}, \boldsymbol{a}_{>j}^K)/d\boldsymbol{a}_j^{k_j}$ can be conducted trivially only if $\mathcal{C}(\boldsymbol{a}_j)$ is empty, otherwise the gradient has to be evaluated recursively using Eq. 21. And this part corresponds to the recursive call in line 11 of Alg. 2.

And the other part of Alg. 2 is the same as Alg. 1. So the rest of the proof follows Thm. 1. Similarly, the Hessian-vector product in line 17 and 18 of Alg. 2 may be approximated as Eq. 20. However, this does not save Alg. 2 from an overall complexity of $\Theta(K^N)$. □

A.2 The Complete Formula for Sec. 3.1 and Sec. 3.2

In this section, we provide the complete formula on $\boldsymbol{y}_i$ related gradient for Sec. 3.1 and Sec. 3.2. Specifically, Eq. 22 is paired with Eq. 10, and Eq. 23 is paired with Eq. 11.

$$\begin{aligned}
\frac{d\mathcal{L}(\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T})}{d\boldsymbol{y}_i} &= \sum_{j=i}^{T} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{y}_i} \\
\frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{y}_i} &= \underbrace{\sum_{l=i+1}^{j} \Big(\frac{\partial \boldsymbol{y}_l}{\partial \boldsymbol{y}_i} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{y}_l} + \frac{\partial \boldsymbol{w}_l}{\partial \boldsymbol{y}_i} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{w}_l} \Big)}_{\text{ignored by BAO}} + \underbrace{\frac{\partial \mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{\partial \boldsymbol{y}_i}}_{\text{considered by BAO}}
\end{aligned} \tag{22}$$

$$\begin{aligned}
\boldsymbol{y}_i^{k_i'+1} \leftarrow \boldsymbol{y}^{k_i'} + \alpha \frac{d\mathcal{L}(\boldsymbol{w}_1^{k_1}, ..., \boldsymbol{w}_i^{k_i}, \boldsymbol{w}_{>i}^K, \boldsymbol{y}_1^{k_1'}, ..., \boldsymbol{y}_i^{k_i'}, \boldsymbol{y}_{>i}^K)}{d\boldsymbol{y}_i^{k_i'}}, \\
\text{where } \boldsymbol{w}_{>i}^0, \boldsymbol{y}_{>i}^0 = f(\boldsymbol{x}, \boldsymbol{w}_1^{k_1}, ..., \boldsymbol{w}_i^{k_i}, \boldsymbol{y}_1^{k_1'}, ..., \boldsymbol{y}_i^{k_i'})
\end{aligned} \tag{23}$$

A.3 An Example of Execution of Alg. 2

In this section, we provide an example of the full procedure of execution of Alg. 2 in Fig. 5. The setup is as Fig. 5.(0): we have $N = 3$ latent $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$ and gradient ascent step $K = 2$, connected by a DAG shown in the figure.

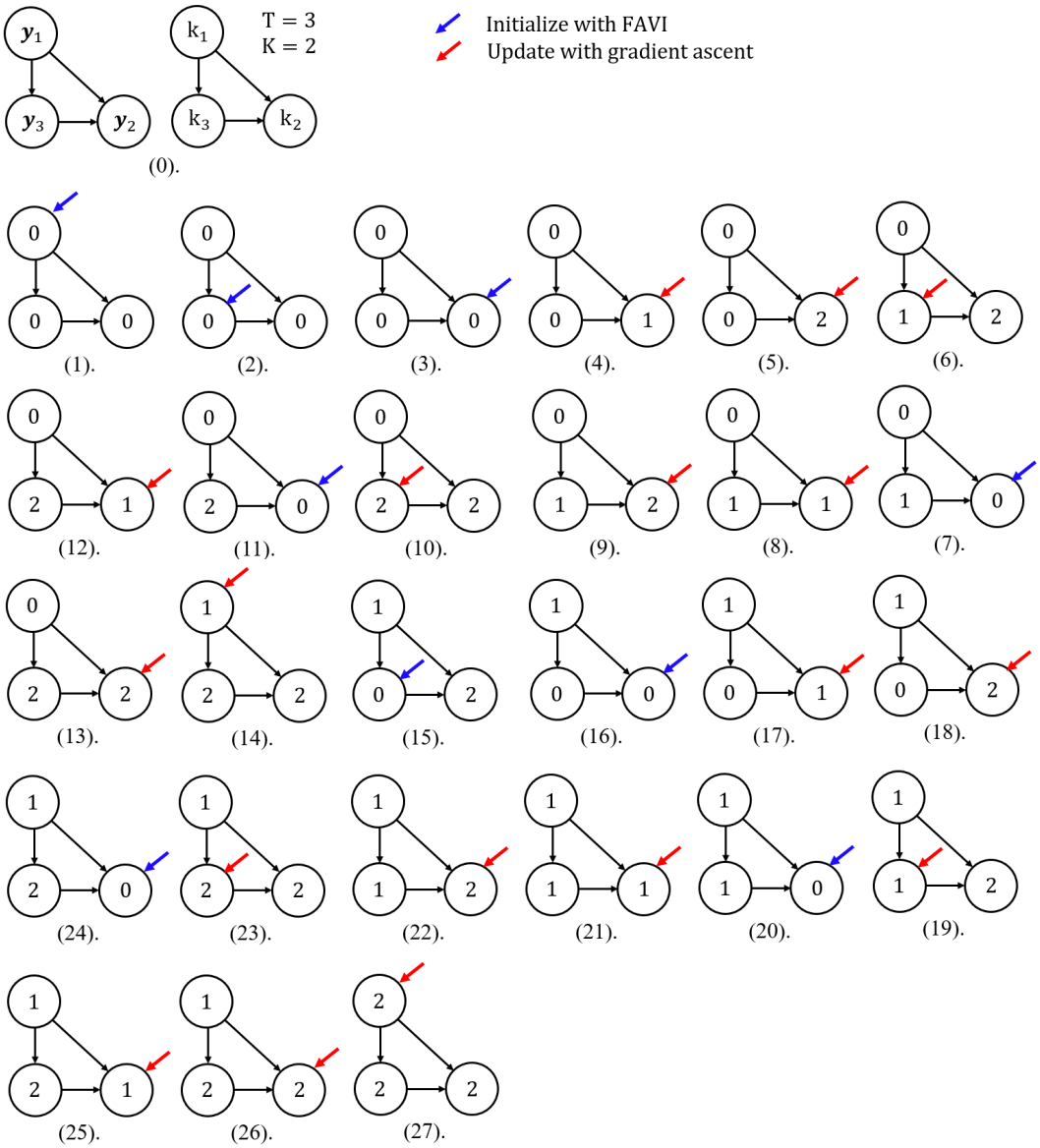

Figure 5: (0). The example setup. (1).-(27). The execution procedure. The number in the circle indicates the gradient step k_i of each node $\boldsymbol{y}_i$. The bold blue/red arrow indicates that the current node is under initialization/gradient ascent.

A.4 Extending the Analysis Sec. 3 to general DAG case

As the Alg. 2 and Alg. 4 are applicable to general SAVI (Kim et al., 2018; Marino et al., 2018) beyond bit allocation, it is helpful to understand their merit to extend the analysis in Sec. 3 from bit allocation to general DAG scenario. In this section, we consider the same problem setup as Sec. 4.2. Similar to bit allocation case, BAO has the gradient incomplete and gradient value incorrect problem. The gradient incomplete issue is presented as Eq. 24, and gradient value incorrect issue is presented

as Eq. 25.

$$\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0},...,\boldsymbol{a}_i^{k_i},\boldsymbol{a}_{>i}^{K})}{d\boldsymbol{a}_i^{k_i}} = \underbrace{\frac{\partial\mathcal{L}(\boldsymbol{a}_0^{k_0},...,\boldsymbol{a}_i^{k_i},\boldsymbol{a}_{>i}^{K})}{\partial\boldsymbol{a}_i^{k_i}}}_{\text{considered by BAO}} + \underbrace{\sum_{\boldsymbol{a}_j\in\mathcal{C}(\boldsymbol{a}_i)}(\sum_{k_j=0}^{K}\frac{\partial\boldsymbol{a}_j^{k_j}}{\partial\boldsymbol{a}_i^{k_i}}\frac{d\mathcal{L}(\boldsymbol{a}_0^{k_0},...,\boldsymbol{a}_{j-1}^{k_{j-1}},\boldsymbol{a}_{\geq j}^{K})}{d\boldsymbol{a}_j^{k_j}})}_{\text{ignored by BAO}} \tag{24}$$

$$\underbrace{\frac{\partial\mathcal{L}(\boldsymbol{a}_0^{k_0},...,\boldsymbol{a}_i^{k_i},\boldsymbol{a}_{>i}^{K})}{\partial\boldsymbol{a}_i^{k_i}} \approx \frac{\partial\mathcal{L}(\boldsymbol{a}_0^{k_i},...,\boldsymbol{a}_i^{k_i},\boldsymbol{a}_{>i}^{k_i})}{\partial\boldsymbol{a}_i^{k_i}}}_{\text{approximation of BAO in gradient value}} \tag{25}$$

A.5 More Implementation Details

In the main text, we use $\boldsymbol{y}_i$ as all the latent variable related to residual. In practice, it is divided into $\boldsymbol{y}_i, \boldsymbol{z}_i, \Delta_i^y$, which refer to the first level latent of residual, second level latent of residual and quantization step size of first level latent of residual respectively. In practice, as BAO (Xu et al., 2022), all of the 3 parts are involved in SAVI jointly. We note that this is not a problem as they fully factorize. And for DVC (Lu et al., 2019), $\boldsymbol{w}_i$ indeed represent the latent of motion. As for DVC, the motion has only one level of latent. However for DCVC (Li et al., 2021), $\boldsymbol{w}_i$ is divided into $\boldsymbol{w}_i, \boldsymbol{v}_i, \Delta_i^w$, which refer to the first level latent of motion, second level latent of motion and quantization step size of first level latent of motion respectively. Similar to $\boldsymbol{y}_i$, all of the 3 parts are involved in SAVI jointly, and this is not a problem as they fully factorize.

Following BAO (Xu et al., 2022), we set the target $\lambda_0 = \{256, 512, 1024, 2048\}$, which also follows the baselines (Lu et al., 2019; Li et al., 2021). We adopt the official pre-train models for both of the baseline methods (Lu et al., 2019; Li et al., 2021). We do not have a training dataset or implementation details for training amortized encoder / decoder as all the experiments are performed on official pre-trained models. For gradient ascent, we set $K = 2000$ for the first I frame and $K = 400$ for all other P frames. On average, the gradient ascent steps for each frame is 920, which is smaller than 2000 in BAO.

A.6 More Quantitative Results

In this section we present more quantitative results. In Tab. 4 we show the BD-PSNR of our proposed method and other methods as a supplementary to the BD-BR results (Tab. 1). Furthermore, in Fig. 6, we present R-D curve on all classes of HEVC CTC and UVG dataset as a supplementary to the HEVC Class D plot (Fig. 2).

	BD-PSNR (dB) ↑				
Method	Cls B	Cls C	Cls D	Cls E	UVG
DVC (Lu et al., 2019) as Baseline					
Li et al. (2016)[1]	-0.54	-	-	-0.32	-0.47
Li et al. (2022)[1]	0.19	-	-	0.28	0.14
OEU (Lu et al., 2020)	0.39	0.49	0.83	0.48	0.48
BAO (Xu et al., 2022)[2]	0.87	1.11	1.17	1.35	0.98
Proposed	1.03	1.38	1.67	1.41	1.15
DCVC (Li et al., 2021) as Baseline					
OEU (Lu et al., 2020)	0.30	0.58	0.74	0.29	0.50
BAO (Xu et al., 2022)[2]	0.52	0.76	0.96	0.96	0.69
Proposed	0.91	1.37	1.55	1.58	1.18

Table 4: The BD-PSNR of our approach compared with baselines (w/o bit allocation) and other bit allocation approaches. [1] comes from Li et al. (2022). [2] comes from (Xu et al., 2022).

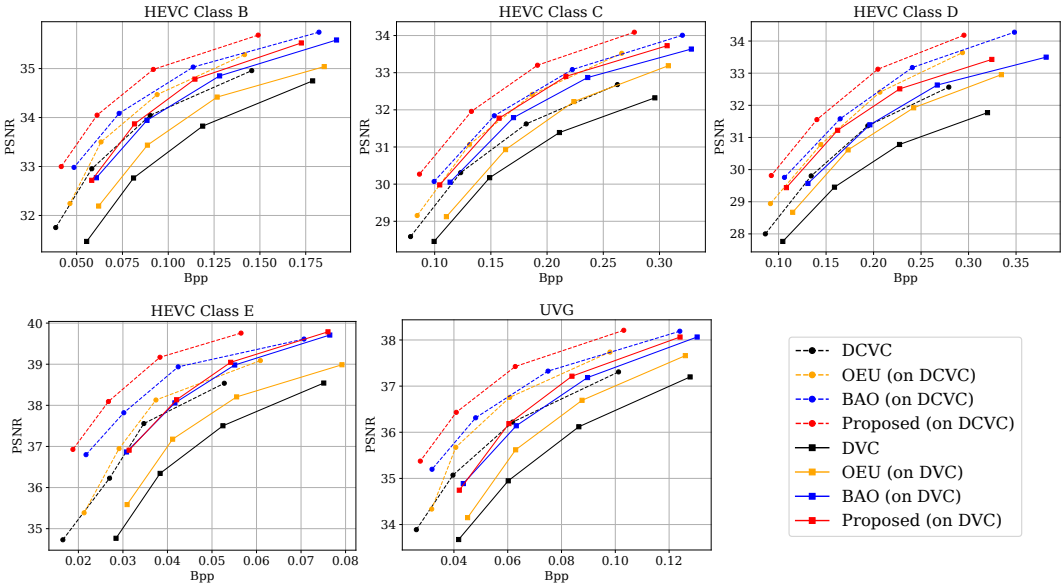

Figure 6: The R-D performance of our approach compared with baselines (w/o bit allocation) and other bit allocation approaches.

A.7 COMPLEXITY & SCALABILITY

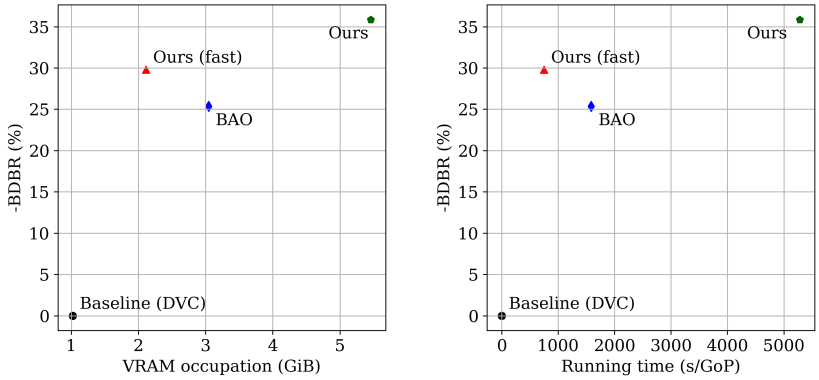

Figure 7: Spatial temporal complexity analysis comparing BAO (Xu et al., 2022), the proposed approach and a fast approximation of the proposed approach. The analysis is done on DVC baseline and HEVC Class D dataset.

We perform additional evaluation to compare the proposed method with BAO (Xu et al., 2022) in terms of temporal complexity and memory cost. The evaluation result can be found in Fig. 7. The general result is that our approach is ≈ 2.8 times slower and cost ≈ 2.0 times memory than BAO, despite the optimization stepsize is smaller. This extra complexity comes from the cost of sequential optimization of latent. And our current method in its naïve form is slower than BAO while performs better. Jointly consider RD performance, time and memory, our method does not dominate BAO.

However, as our approach enables a sequential style semi-amortized variational inference (SAVI) (Kim et al., 2018; Marino et al., 2018) on latents, there exists a very simple trick to speed it up. Moreover, this trick also resolves the scalability issue. Specifically, to optimize the i^{th} frame's latent, we do not compute the R-D cost of all the frames after it as we do now. Instead, we limit the

R-D cost computation to a small fixed size of frames. Formally, we approximate the gradient as:

$$\frac{d\mathcal{L}(\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T})}{d\boldsymbol{w}_i} \approx \sum_{j=i}^{i+C} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{w}_i}, \frac{d\mathcal{L}(\boldsymbol{w}_{1:T}, \boldsymbol{y}_{1:T})}{d\boldsymbol{y}_i} = \sum_{j=i}^{i+C} \frac{d\mathcal{L}_j(\boldsymbol{w}_{1:j}, \boldsymbol{y}_{1:j})}{d\boldsymbol{y}_i} \tag{26}$$

, where C is a preset constant indicating the number of future frames we included for consideration. With this trick, our algorithm approach cost only $\approx 50\%$ of time and $\approx 60\%$ of memory compared with BAO, while remains a superior performance ($\approx 5\%$ better in BDBR) (Ours (fast) in Fig. 7, the results are based on DVC Class D $c = 2$). With this trick, jointly consider RD performance, time and memory, our approach clearly dominates BAO. Furthermore, with this trick, the scalability issue of

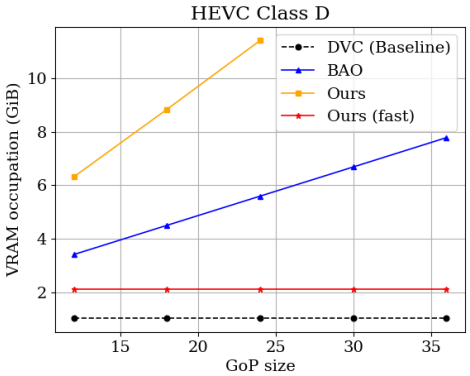

Figure 8: Scalability analysis comparing BAO (Xu et al., 2022), the proposed approach and the fast approximation of the proposed approach. The analysis is done on DVC baseline and HEVC Class D dataset.

our approach is significantly ellivated. As shown in Fig. 8, the memory cost our approach with this trick is constant to GoP size, while that of BAO and our approach without this trick grows linearly with GoP size. This means that with this trick, our approach becomes scalable to any GoP, which is superior than BAO.

A.8 Impact on OEU

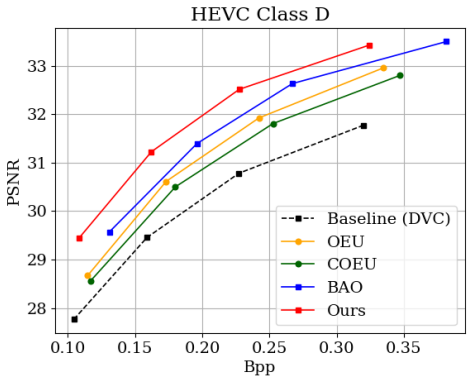

Figure 9: Replace the joint update of OEU (Lu et al., 2020) by our sequential update (Alg. 4). The analysis is done on DVC baseline and HEVC Class D dataset.

Another interesting question to ask is whether the sequential updating algorithm (Alg. 4) benefits the OEU (Lu et al., 2020). Indeed, OEU (Lu et al., 2020) and BAO (Xu et al., 2022) are quite similar at the first glance. However, it is important to note that the theoretical foundation of BAO and this paper is SAVI (Kim et al., 2018; Marino et al., 2018). However, OEU does not fit into SAVI

framework. More specifically, its encoder parameter to be updated does not factorizes as the DAG defined by variational posterior. Thus, applying Alg. 4 is incorrect. To verify this empirically, we change the OEU from BAO's joint optimization to ours sequential optimization (Alg. 4), and the results show that this change degrades R-D performance (See COEU line in Fig. 9).

A.9 More Analysis

In this section, we extend the analysis on why the proposed approach works and what is the difference between the proposed approach and BAO (Xu et al., 2022). In the approximate SAVI on DAG latent (Alg. 4), we solve SAVI approximately latent by latent in topological order. For bit allocation of NVC with 10 frames, this topological order is $\boldsymbol{y}_0, \boldsymbol{w}_1, \boldsymbol{y}_1, ..., \boldsymbol{w}_9, \boldsymbol{y}_9$, where $\boldsymbol{y}_0$ is the latent of I frame, $\boldsymbol{w}_i$ is the motion latent of i^{th} P frame and $\boldsymbol{y}_i$ is the residual latent of i^{th} P frame. In Fig. 10, we show the relationship between R-D cost and the stage of approximate SAVI. We can see that the R-D cost reduces almost consistently with the growing of SAVI stage, which indicates that our approximate SAVI on DAG (Alg. 4) is successful. Specifically, despite our approach is inferior to BAO (Xu et al., 2022) upon the convergence of $\boldsymbol{y}_3$, it attains significant advantage over BAO after $\boldsymbol{y}_9$ converges.

In Fig. 11, we compare the distribution of R-D cost, PSNR and Bpp across frame and latent of the baseline DVC (Lu et al., 2019), BAO Xu et al. (2022) and the proposed approach. For R-D cost, it is obvious that our proposed approach's R-D cost is lower than BAO and baseline, which indicates a better R-D performance. For bpp, it is interesting to observe that despite all three methods have similar bpp of motion related latent $\boldsymbol{w}_{1:T}$, the bpp of residual related latent $\boldsymbol{y}_{1:T}$ is quite different. Specifically, BAO increases the bpp of $\boldsymbol{y}_{1:T}$ compared with baseline, while our approach decreases the bpp of $\boldsymbol{y}_{1:T}$ compared with baseline. This explains why our approach has lower bitrate compared with BAO, and also explains why our approach has significantly less bitrate error. For the PSNR metric, both our approach and BAO significantly improve the baseline. And the difference between proposed approach and BAO is not obvious. We can conclude that the benefits of the proposed approach over BAO comes from the bitrate saving instead of quality enhancing.

A.10 Qualitative Results

In Fig. 12, Fig. 13, Fig. 14 and Fig. 15, we present the qualitative result of our approach compared with the baseline approach. 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. We intentionally omit the qualitative comparison with BAO (Xu et al., 2022) as it is not quite informative. Specifically, from Fig. 2 we can observe that the PSNR difference of BAO and our approach is very small (within ± 0.1dB). And our main advantage over BAO comes from bitrate saving instead of quality improvement. Thus the qualitative difference between the proposed method and BAO is likely to fall below just noticeable difference (JND).

A.11 More Discussion

Other weakness includes scalability. Our method requires jointly considering all the frame inside the GoP, which is impossible when the GoP size is large or when GoP size is unknown for live streaming tasks. Furthermore, currently the gradient ascent step number is merely chosen as an empirical sweet spot between speed and performance. A thorough grid search is desired to better understand its effect on performance.

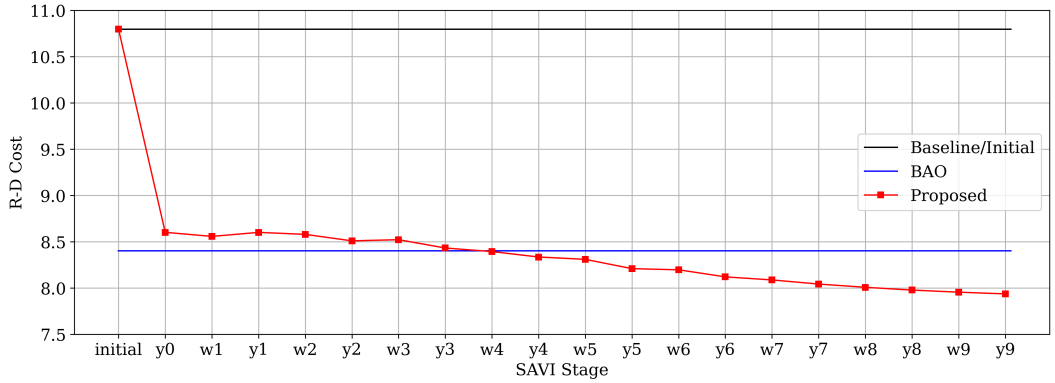

Figure 10: The R-D cost versus SAVI Stage. Experiment is conducted with DVC (Lu et al., 2019) as baseline and *BasketballPass* of HEVC Class D as data.

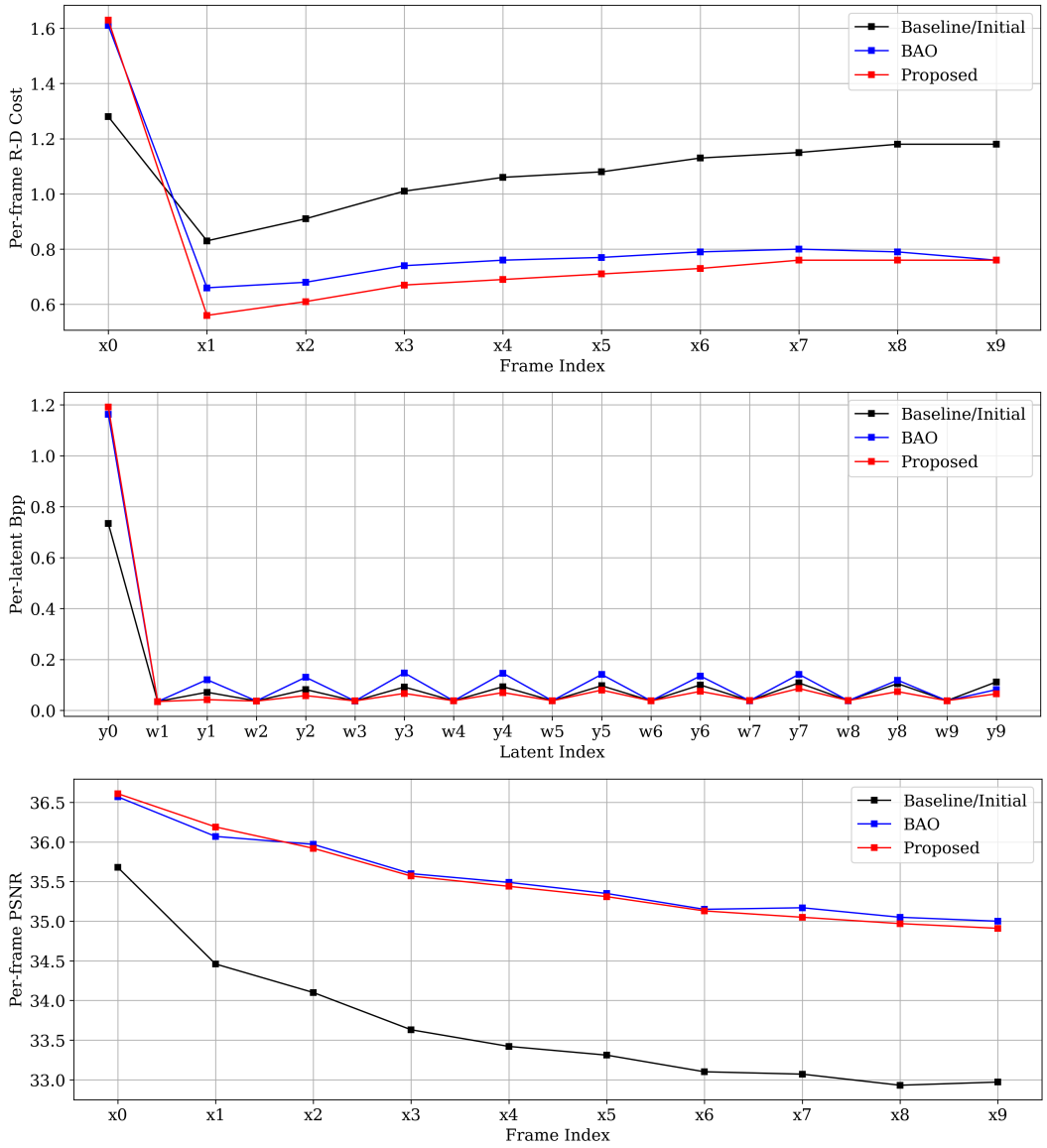

Figure 11: The R-D cost, bpp and PSNR versus frame/latent index. Experiment is conducted with DVC (Lu et al., 2019) as baseline and *BasketballPass* of HEVC Class D as data.

Figure 12: Qualitative results using *BasketballPass* of HEVC Class D. *Top*. Original frame. *Middle*. Baseline codec (DVC)'s reconstruction result with bpp $= 0.148$ and PSNR $= 33.06$dB. *Bottom*. Proposed method's reconstruction result with bpp $= 0.103$ and PSNR $= 34.91$dB.

Figure 13: Qualitative results using *BlowingBubbles* of HEVC Class D. *Top*. Original frame. *Middle*. Baseline codec (DVC)'s reconstruction result with bpp $= 0.206$ and PSNR $= 30.71$dB. *Bottom*. Proposed method's reconstruction result with bpp $= 0.129$ and PSNR $= 32.34$dB.

Figure 14: Qualitative results using *BQSquare* of HEVC Class D. *Top*. Original frame. *Middle*. Baseline codec (DVC)'s reconstruction result with bpp $= 0.232$ and PSNR $= 28.72$dB. *Bottom*. Proposed method's reconstruction result with bpp $= 0.128$ and PSNR $= 30.87$dB.

Figure 15: Qualitative results using *RaceHorses* of HEVC Class D. *Top*. Original frame. *Middle*. Baseline codec (DVC)'s reconstruction result with bpp $= 0.448$ and PSNR $= 30.48$dB. *Bottom*. Proposed method's reconstruction result with bpp $= 0.379$ and PSNR $= 31.92$dB.