Efficient In-situ Image and Video Compression through Probabilistic Image Representation

Rongjie Liu[a,*], Meng Li[b], Li Ma[c]

[a]*Department of Statistics, Florida State University, Tallahassee, FL, USA*
[b]*Department of Statistics, Rice University, Houston, TX, USA*
[c]*Department of Statistical Science, Duke University, Durham, NC, USA*

Abstract

Fast and effective image compression for multi-dimensional images has become increasingly important for efficient storage and transfer of massive amounts of high-resolution images and videos. In this paper, we present an efficient in-situ method for multi-dimensional image and video compression called Compression via Adaptive Recursive Partitioning (CARP). CARP uses an optimal permutation of the image pixels inferred from a Bayesian probabilistic model on recursive partitions of the image to reduce its effective dimensionality, achieving a parsimonious representation that preserves information. Furthermore, it adopts a multi-layer Bayesian hierarchical model to achieve in-situ compression along with self-tuning and regularization, with just one single parameter to be specified by the user to achieve the desired compression rate. The properties of our proposed method include high reconstruction quality at a wide range of compression rates while preserving key local details, applicability to a variety of different image/video types and of different dimensions, computational scalability, progressive transmission and ease of tuning. Extensive numerical experiments using a variety of datasets including 2D still images, real-life YouTube videos, and surveillance videos show that CARP compares favorably to—and often uniformly outperforms—a wide range of popular image/video compression approaches, including JPEG, JPEG2000, AVI, BPG, MPEG4, HEVC, AV1, and

*Corresponding author
Email address: rliu3@fsu.edu (Rongjie Liu)

three neural network-based methods.

Keywords: Image compression, video compression, Bayesian probabilistic model, in-situ compression

1. Introduction

Image compression is the key to efficient storage and transfer of the vast amount of high-resolution images and videos that are routinely collected in a variety of applications. Efficient compression relies on parsimonious representations of images that preserve important spatial and contextual features. Standards such as JPEG [1] and JPEG2000 [2] utilize fixed, deterministic linear function transforms, such as wavelets followed by optimized encoding under such transforms. BPG [3], short for Better Portable Graphics, is based on the intra-frame encoding of the High Efficiency Video Coding (HEVC) video compression standard and has also been widely used in image compression as an alternative to JPEG in some aspects. These approaches give excellent stability and scalability in practical implementation, and require little training and tuning. However, they lack adaptability to image-specific characteristics, consequently resulting in a suboptimal level of compression efficiency. The more recent convolutional neural network (CNN) based approaches [4, 5, 6] utilize much more flexible, nonlinear transformations of the original image. This additional flexibility often leads to improved compression efficiency. On the other hand, this also results in a considerably heightened demand for intensive training and meticulous method refinement. Driven by the identified limitations of existing methodologies, the primary objective of this paper is to devise a compression technique capable of alleviating the inherent constraints posed by deterministic transforms. Our aim is to achieve cutting-edge performance without the necessity for separate training procedures, thereby optimizing both computational and compression efficiency.

The primary contribution of this paper lies in the introduction of a Bayesian probabilistic modeling approach. This approach incorporates adaptivity to im-

age features within the framework of wavelet-based image processing while preserving computational scalability and ease of tuning. Instead of employing fixed wavelet transforms, we consider the transform as an unobservable element of interest. We achieve this by imposing a Bayesian prior over the space of such transforms, generated through random recursive partitioning on the image. Consequently, a compressed image can be generated based on the inferred transform, tailored to the specific features of the image in question. Furthermore, this computational process remains as efficient as traditional wavelet-based methods, exhibiting linear scaling in relation to the image's size. Aside from achieving excellent compression efficiency (to be demonstrated in our numerical experiments), our method, called Compression via Adaptive Recursive Partitioning (CARP), enables simultaneous transform learning and compression through obtaining the joint posterior distributions of the transform and intensities of the images, allowing information borrowing between these two steps, which is in contrast to classical methods such as JPEG2000. CARP is model-based and interpretable. Building on hierarchical Bayesian models, CARP outputs the *maximum a posteriori* (MAP) partition tree-induced transform, which represents a learned map to vectorize the image.

CARP enjoys three additional advantages. First, CARP is a general-purpose compression method that compresses m-dimensional images in a unified fashion for $m = 2, 3, 4$, etc., as opposed to specializing on one dimension (such as 2D or 3D) in most of the existing literature, making it readily applicable to a variety of image/video types. Second, it does not require a separate training stage on external data and involves minimal tuning. The Bayesian hierarchical modeling strategy uses hyperpriors on the parameters to allow automatic tuning on those parameters, leaving only one free parameter for the user to specify, which corresponds directly to the desired compression rate of the image. This makes CARP very easy to use, without requiring expert knowledge of the underlying method. Third, it allows progressive transmission to gradually improve the image quality as data bits are incrementally transmitted, which resembles the progressive principle in JPEG2000 [7]. In particular, the wavelet coefficients

can be transmitted from coarse to fine scales in the decoding and reconstruction process to allow reconstruction of images at incremental resolutions. For example, Figure 1 demonstrates that the reconstructed building image is gradually improved when more data bits are transmitted.

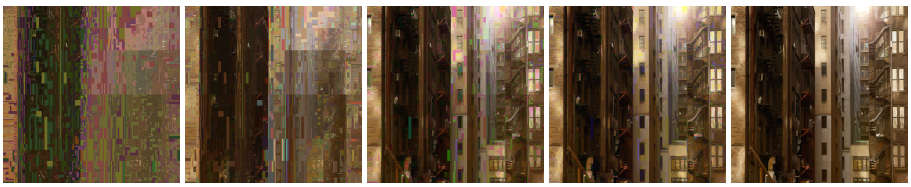


Figure 1: The reconstructed image is gradually improved from left to right when more data bits are transmitted (left to right: 300, 3000, 30000, 40000, 50000 bits). The original image is in Figure 5.

This paper is an extended version of our previous work [8]; we have added technical descriptions for our probability model and inference strategy in Sections 3.4 and 3.5, improved the method by allowing flexible encoding/decoding algorithm options that adapt to the resolution of target images, elaborated on properties of CARP such as progressive transmission, substantially expanded our numerical experiments by both including a new color image database and several other competing approaches, including AVI, BPG, HEVC, AV1 and another two deep learning-based methods in Section 4, and discussed a range of practical issues and options to adapt CARP in Section 5. Of particular note is that the improved CARP exhibits noticeable performance gain over our initial implementation in [8] under the metric of peak signal-to-noise ratio for low-resolution images/videos (see Figure S4 in the Supplementary Material).

The organization of the paper is as follows. Section 2 surveys related work in image and video compression. In Section 3, we present the proposed CARP method, including probabilistic image representation using a hierarchical Bayesian framework, posterior inference, and details in implementing the method. Section 4 reports the results from extensive experiments on image and video compression. Section 5 contains a discussion. The code for average performance and

implementing CARP is available on GitHub: `https://github.com/xylimeng/CARP`. Supplementary Materials include additional visualization for experiments in Section 4.

2. Related work

For 2D images, two of the most well-known compression algorithms are JPEG [1] and its successor JPEG2000 [2]. The JPEG standard uses a discrete cosine transform (DCT) on each 8 by 8 small block of pixels. A quantization table is applied, and Huffman encoding is used on DCT blocks for compression. Compared to the JPEG standard, JPEG2000 uses a multi-scale orthogonal wavelet decomposition with arithmetic coding. In particular, the discrete wavelet transform (DWT) decomposes images into their resolution and frequency contents. The DWT can be performed with a non-reversible Daubechies (9,7) taps filter, which provides higher, but lossy, compression. In addition to DWT, another feature for JPEG2000 is quantization. JPEG2000 quantizer follows an embedded dead-zone scalar approach. The quantizer step size used to scale the coefficients is independently selected for each wavelet sub-band.

However, both JPEG and JPEG2000 are suboptimal for image compression [9] in part due to the non-adaptive image transformation and a separate optimization on codecs. BPG [3] is often advantageous over JPEG and JPEG2000 in achieving better compression ratios for the same reconstruction quality, and it supports up to 14 bits per color channel instead of up to 8 bits as in JPEG.

Besides JPEG, JPEG2000 and BPG, there is a growing literature on developing deep learning-based methods [4, 5, 10] for image compression. Among these methods, end-to-end deep learning-based approaches are particularly appealing, which go directly from the input to the desired output with optimized codecs [5]. For example, a pre-trained model over a database of training images was proposed in [5] with all the required components for end-to-end implementation, including a nonlinear analysis transformation, a uniform quantizer, and a nonlinear synthesis transformation. On this basis, two pre-trained models are

developed (Sadam and GDN) in the context of nonlinear transform coding with artificial neural networks [10].

Videos have a different structure than 2D images due to the extra temporal dimension. Although a video can be compressed frame by frame by some existing 2D image compression methods (e.g., JPEG, JPEG2000, and BPG), the critical temporal redundancy is undesirably ignored in this approach. A classical video compression format is AVI (Audio Video Interleave), which was created by Microsoft in 1992 and becomes a commonly used video format. AVI is internet compatible but generally suffers from low compression efficiency by current standards. Most current video compression algorithms in Moving Picture Experts Group (MPEG) [11] exploit both spatial and temporal redundancy and deal with the issue of heavy compression ratio. For example, MPEG-4 absorbs many features of different standards using both DCT and motion compensation [12] techniques to achieve this goal. In addition, to reach a higher compression ratio, MPEG-4 only stores and encodes the inter-frame changes instead of the entire original frame. However, the redundancy detection strategy in MPEG-4 is localized to capturing the difference of adjacent frames, and thus might not be globally optimal, leading to less efficient compression. HEVC [13], standing for High Efficiency Video Compression, is an extension of MPEG-4 designed by an intra-frame coding strategy of applying intra-picture prediction and loop filters to optimally use parallel processing and improve the quality of the reconstruction frames. AV1 (AOMedia Video 1) is a more recent codec designed to compete with HEVC on the open platform.

The idea of transform-induced permutations of the pixels employed in CARP has been exploited previously in the literature. In particular, [14] adopts peak transform to obtain spatial permutation. [15] uses random recursive partitioning to induce a probability distribution on the permutations of image pixels, leading to an effective algorithm for image denoising using posterior Bayesian model averaging. In this work we use random recursive partitioning to induce a probability model on the wavelet transforms, but instead focus on learning a data-adaptive transform to represent the image, thereby achieving efficient com-

pression. The 1D vector returned by Bayesian multi-scale learning is passed to existing encoding methods to generate compressed representation, followed by the corresponding decoder. The literature [16] provides a rich menu of possibilities for encoding/decoding methods.

3. Method

3.1. CARP: The framework

CARP is a framework for image compression via adaptive recursive partitioning. It uses a data-adaptive permutation of the image pixels inferred from a Bayesian probabilistic model to reduce the dimensionality of an m-dimensional image, thereby achieving a parsimonious representation that effectively preserves information. More specifically, CARP utilizes a prior distribution on the space of permutations induced by random recursive partitioning along a bifurcating tree [15]. This random recursive partitioning incorporates latent pruning variables to probabilistically terminate the partitioning within the partition blocks where the pixel intensities are similar enough. The *maximum a posteriori* (MAP) estimate, i.e., the posterior mode of the posterior distribution on the recursive partitioning, produces a representative permutation (or vectorization) of the image pixels that can be readily fed into encoding methods to generate compressed representation, followed by the corresponding decoder to reconstruct the compressed image.

Figure 2 presents the architecture of CARP, where the two black boxes pinpoint the key techniques used in CARP. In this work, we use the 1D discrete wavelet transform (DWT) and either Arithmetic or Huffman encoding algorithm as the encoder, and use the inverse DWI and corresponding decoding algorithm as the decoder.

CARP takes an m-dimensional image $\boldsymbol{y} = \{y(\boldsymbol{x}) : \boldsymbol{x} \in \Omega\}$ observed on an m-dimensional rectangular "pixel" space Ω of the form

$$\Omega = [0, n_1 - 1] \times [0, n_2 - 1] \times \cdots \times [0, n_m - 1],$$

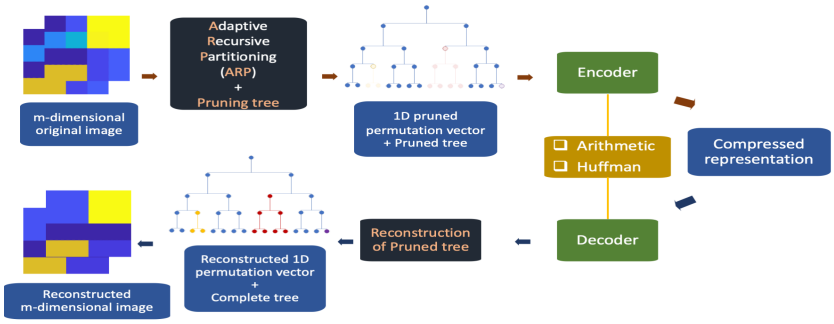


Figure 2: The workflow of CARP.

where the notation $[a, b]$ is the set $\{a, a+1, \ldots, b\}$ for two integers a and b with $a \leq b$. This means CARP can be readily applied to images of various dimensions including but not limited to 2D still images and 3D videos. Without loss of generality, we assume $n_i = 2^{J_i}$ in the ith dimension for $i = 1, 2, \ldots, m$; an image of general size can be upsized to such dimensions through padding. The total number of pixels is $n = 2^J$, where $J = \sum_{i=1}^{m} J_i$.

The effectiveness of the compression highly depends on the representation power of the transform in use, especially on its adaptivity to local and spatial features in an image. In CARP, this is achieved by a Bayesian probabilistic modeling strategy, in which adaptivity to image features is incorporated into a wavelet transform-based multi-scale image processing framework. In particular, we use random recursive partitioning on Ω to induce models on wavelet transforms that incorporate such adaptivity. In the following section, we describe some basic concepts related to recursive dyadic partitioning, which will form the building blocks for the model used in CARP.

3.2. Recursive dyadic partitioning with pruning

While multi-scale wavelet transforms enjoy excellent scalability, a deterministic transform may fail to efficiently adapt to the rich spatial and local features present in a multi-dimensional image. We enrich the representation power and

effectiveness of wavelets by a convolution between a classic 1D wavelet transform and a class of permutation of the index space Ω of the pixels.

The space consisting of all permutations of pixels in Ω is massive. Considering all $n!$ permutations of pixels in Ω is not only computationally prohibitive but wasteful as well because the vast majority of permutations ignore the spatial features in the image. In CARP, we only consider the class of permutations induced by a recursive dyadic partitioning (RDP) on Ω, which includes a rich class of permutations for effective representation of the image while allowing scalable learning of the optimal permutation among this class—with computational complexity $O(n)$. An RDP on Ω denoted by $\mathcal{T}$ as it is essentially a bifurcating tree, consists of a sequence of nested partitions on Ω, i.e., $\mathcal{T} = \cup_{j=0}^{J} \mathcal{T}^j$ with the partition $\mathcal{T}^j$ being the set of all blocks at level j for $j = 0, \ldots, J$. Specifically, we start with $A_{0,0} = \Omega$ and $\mathcal{T}^0 = \{A_{0,0}\}$. For each $j = 0, \ldots, J-1$, $\mathcal{T}^{j+1}$ is obtained by dividing each set in $\mathcal{T}^j$ into two halves along a divisible dimension, i.e., $A_{j,k} = A_{j+1,2k} \cup A_{j+1,2k+1}$ for $k = 0, 1, \ldots, 2^j - 1$. The last level $\mathcal{T}^J$ contains all the single elements in Ω, which are referred to as *atomic* nodes.

From now on we shall refer to the partition blocks as "nodes" in the partition tree $\mathcal{T}$. Two children nodes are formed by dividing a parent node into two halves in one of its dimensions, and $\mathcal{T}^J$ consists of the leaf nodes, each of which contains a single element in Ω. Note that each RDP induces a unique permutation of Ω, with the order of the pixels given by the binary coding sequence tracking the left/right children that each pixel belongs to along the corresponding branch in the tree.

Parsimonious representation is crucial for image compression. We improve upon RDPs in this regard by incorporating early stopping to prune the partition tree induced by an RDP. Indeed, a complete RDP $\mathcal{T}$ might not be necessary to represent an image when there are homogeneous nodes with almost constant intensities therein. In CARP, we prune $\mathcal{T}$ by an early stopping to help effectively compress similar blocks in an image. Note that all descendants of a pruned node are pruned by design, and the intensity of each pixel in the corresponding block will be set to the average value of this block.

Each RDP turns the pixel space Ω into a vector of the same length as the number of pixels, decoupling the image intensity to a 1D vector and spatial structure described by a partition tree with pruning. We next describe generative models for RDPs and the image given RDPs.

3.3. Generative modeling of RDPs

We use a hierarchical Bayesian model to adaptively learn a representative RDP through finding the posterior mode. To this end, we adopt a generative distribution called "random RDP" (RRDP) proposed in [15] as the prior on the RDP. While RRDP has been successfully applied to various tasks such as testing of conditional association and image reconstruction, there is no work to consider using RRDP for compression. For any node A in $\mathcal{T}$, $\boldsymbol{\lambda}$ denotes a mapping from A to a hyperparameter $\boldsymbol{\lambda}(A)$ that specifies the probability to divide A in each of its divisible dimensions. RRDP specifies the probability of partitioning A in its ith dimension for $i = 1, \ldots, m$ using a vector-valued hyperparameter $\boldsymbol{\lambda}(A) = (\lambda_1(A), \ldots, \lambda_m(A))$. By default, we set the value of $\boldsymbol{\lambda}$ such that all divisible dimensions of each A have equal prior probability to be divided.

3.4. Data generating model given RDPs: Markov-tree wavelet regression

To achieve a parsimonious image representation, the next component of our model aims at pruning the tree in nodes where the pixel intensities are similar enough. To this end, we use wavelet shrinkage. In particular, given an RDP, we adopt a wavelet regression model as the data generative mechanism for the image. There is a rich literature on how to effectively carry out shrinkage on the wavelet coefficients [17, 18], and we shall use a Bayesian wavelet regression model with a Markov tree prior on the wavelet coefficients to achieve adaptive shrinkage [15]. An important benefit of adopting a Bayesian model for the image given the RDP is that we can now combine it with our Bayesian model on the RDPs to form a coherent hierarchical model, allowing inference to be carried out

in a principled manner (through maximizing the posterior distribution) without *ad hoc* strategies to "stitching" together separate algorithmic pieces.

Specifically, conditional on an RDP tree $\mathcal{T}$ and following an application of Haar wavelet transform to the vectorized image under $\mathcal{T}$, the Bayesian wavelet regression model is as follows

$$w_{j,k} = z_{j,k} + u_{j,k} \tag{1}$$

$$z_{j,k} \,|\, S_{j,k} \overset{\text{ind}}{\sim} \begin{cases} \delta_0(\cdot) & \text{if } S_{j,k} = 0 \text{ or } 2 \\ \text{Normal}(0, \tau_j^2 \sigma^2) & \text{if } S_{j,k} = 1 \end{cases} \tag{2}$$

for $j = 0, 1, \ldots, J-1$ and $k = 0, 1, \ldots, 2^j - 1$. Here $w_{j,k}$, $z_{j,k}$, $u_{j,k}$ are the kth wavelet coefficient, signal, and "noise" at the jth scale in the wavelet domain, respectively. The ternary latent state variable $S_{j,k}$ indicates whether $z_{j,k}$ is from $\delta_0(\cdot)$ (a point mass at 0) if $S_{j,k} = 0$ or 2, or a normal distribution with mean 0 and variance $\tau_j^2 \sigma^2$ if $S_{j,k} = 1$.

To achieve adaptive pruning, we model $S_{j,k}$ jointly by a Markov tree model [19] such that if $S_{j-1,\lfloor k/2 \rfloor} = 2$ then $S_{j,k} = 2$ with probability 1. Thus $S_{j,k} = 2$ is an "absorbing state" representing the pruning of a branch of $\mathcal{T}$. If $S_{j-1,\lfloor k/2 \rfloor} \neq 2$ then $S_{j,k} = (0, 1, 2)$ with probabilities $(\rho(A_{j,k})\{1 - \lambda_0(A_{j,k})\}, \{1 - \rho(A_{j,k})\}\{1 - \lambda_0(A_{j,k})\}, \lambda_0(A_{j,k}))$, where $\rho(A)$ is the so-called spike probability in Bayesian spike-and-slab models, and $\lambda_0(A)$ is the probability of node A to be pruned. We assume $u_{j,k} \sim N(0, \sigma^2)$ independently across j and k.

It is worth noting that in the context of compressing noiseless images, the "noise" term $u_{j,k}$ quantifies the extent of local variations in pixel intensities to which one can ignore to produce a compressed image, and therefore its standard deviation σ becomes a parameter for setting how aggressively (in terms of the compression ratio) one wants to compress the image through pruning the tree. This tuning parameter σ will be in a monotone relation to the final compression rate of the image, and thus can be set by the user to achieve the desired rate of compression.

3.5. Posterior inference and image representation

For image compression, we need to find a single representative RDP that most effectively represents features in the image. One strategy is to draw a random sample of $\mathcal{T}$ from its posterior distribution, and use the induced permutation for compression. However, a more appealing approach is to resort to a particular non-random representative sample. To this end, we maximize the posterior probability of $\mathcal{T}$ based on its marginal posterior distribution. In other words, we aim to find the *maximum a posteriori* (MAP) estimate for $\mathcal{T}$, which we denote as $\hat{\mathcal{T}}$.

We first need to find the marginal posterior distribution for $\mathcal{T}$. A remarkable observation is that under the above model, the posterior of $\mathcal{T}$ is conjugate—it is still an RRDP distribution with pruning, but with updated posterior selection probabilities $\tilde{\boldsymbol{\lambda}}$ and updated pruning probabilities $\tilde{\lambda}_0$, where we use tilde to indicate the posterior updated values for the parameters $\boldsymbol{\lambda}$ and λ_0. Such conjugacy can be obtained in a general Markov tree setting as long as the joint distribution of $(w_{j,k}, z_{j,k})$ given $S_{j,k}$ is independent for all j and k, which has been studied in Theorem 2 of [15]. We next derive such conjugacy specifically for our model in (1) and provide recipes for inference on $\mathcal{T}$.

Let $\mathcal{A}$ be the set collecting the nodes generated by all possible RDPs. Let $\Psi(A)$ be the marginal likelihood of node A for $A \in \mathcal{A}$. Hereafter by the marginal likelihood "of a node" we refer to the marginal likelihood from all data with locations in A if the parent node of A is not pruned (i.e., integrating out those $\mathcal{T}$'s that contain the node A with respect to their prior distribution). We start with introducing a few quantities associated with each node A that will be used for describing the posterior. Let $D(A)$ be the set collecting all divisible dimensions of A, $y(A) = \sum_{\boldsymbol{x} \in A} y(\boldsymbol{x})$ the sum of observations with locations in A, and $\mathrm{SST}(A)$ the corrected sum of squares of all data with locations in A, i.e., $\mathrm{SST}(A) = \sum_{x \in A}(y(\boldsymbol{x}) - y(A)/|A|)^2$ with $|A|$ being the number of locations in A. For each $d \in D(A)$ such that A is divided in its dth dimension, let $(A_l^{(d)}, A_r^{(d)})$ be the pair of left and right children nodes, and $w_d(A) = \{y(A_l^{(d)}) - y(A_r^{(d)})\}/\sqrt{|A|}$ be the Haar wavelet coefficient on A.

The marginal likelihood $\Psi(A)$ can be obtained in a recursive manner with complexity $O(n)$. To see this, first decompose $\Psi(A)$ according to the $|D(A)|+1$ possible actions on A:

$$\Psi(A) = \lambda_0 \Psi_0(A) + (1-\lambda_0) \sum_{d \in D(A)} \lambda_d \Psi_d(A), \tag{3}$$

where $\Psi_d(A)$ is the marginal likelihood of A if A is divided in its dth dimension given A is not pruned, and $\Psi_0(A)$ is the marginal likelihood of A if A is pruned. When $d = 0$, all wavelet coefficients in A and its descendants have mean 0, and thus

$$\Psi_0(A) = (2\pi\sigma^2)^{-\frac{|A|-1}{2}} \exp\left\{-\frac{\mathrm{SST}(A)}{2\sigma^2}\right\}. \tag{4}$$

When $d \in D(A)$, $\Psi_d(A)$ is independently decomposed into the marginal likelihood of $w_d(A)$ that is spelled out in (1) and the marginal likelihoods of its two children nodes, that is,

$$\begin{aligned}\Psi_d(A) =& \{\rho(A) N(w_d(A); 0, (1+\tau_j^2)\sigma^2) + (1-\rho(A)) \\ & N(w_d(A); 0, \sigma^2)\} \Psi(A_l^{(d)}) \Psi(A_r^{(d)}),\end{aligned} \tag{5}$$

where $N(a; b, c^2)$ is the density function of Normal(b, c^2) evaluated at a. A bottom-up algorithm is then readily applicable to obtain all $\Psi(A)$'s by combining (3), (4), and (5) and letting $\Psi(A) = 1$ for all atomic nodes A.

Once $\Psi(A)$ is calculated, a direct application of Bayes' theorem gives $\tilde{\lambda}_0(A) = \lambda_0(A)\Psi_0(A)/\Psi(A)$ and $\tilde{\lambda}_d(A) = \lambda_d(A)\Psi_d(A)/\{\sum_{d \in D(A)} \lambda_d(A)\Psi_d(A)\}$, which respectively are the posterior probability of pruning A and the posterior probability for A to be divided in dimension d if A is not pruned. These two mappings completely describe the marginal posterior distribution of $\mathcal{T}$. We then compute the MAP tree $\hat{\mathcal{T}}$ by standard bottom-up dynamic programming as follows, which again incurs complexity $O(n)$. Define a recursive mapping $\kappa : \mathcal{A} \to [0, 1]$:

$$\kappa(A) = \begin{cases} 1, & \text{if A is atomic,} \\ \max\{\tilde{\lambda}_0(A), \tilde{\lambda}_{-0}^{\max}(A)\}, & \text{otherwise,} \end{cases}$$

where

$$\tilde{\lambda}_{-0}^{\max}(A) = (1 - \tilde{\lambda}_0(A)) \max_{d \geq 1} \{\tilde{\lambda}_d(A) \kappa(A_l^{(d)}) \kappa(A_r^{(d)})\}.$$

Once the mapping κ has been computed on all $A \in \mathcal{A}$, $\hat{\mathcal{T}}$ can be generated inductively as follows. Note that $\hat{\mathcal{T}}$ consists of partition index sets $\{\hat{A}_{j,k} : 0 \le k < 2^j, 0 \le j \le J\}$ and a pruning indicator vector associated with each $\hat{A}_{j,k}$. For $j = 0$, $\hat{A}_{0,0} = \Omega$. Suppose $\hat{\mathcal{T}}$ have been generated for all levels no more than j. For each $\hat{A}_{j,k}$ that is not pruned, define $\hat{d}_{j,k} = \text{argmax}_{d \ge 1} \tilde{\lambda}_d(A)\kappa(A_l^{(d)})\kappa(A_r^{(d)})$ with $A = \hat{A}_{j,k}$. Then, block $A_{j,k}$ is pruned if $\tilde{\lambda}_0(A) > (1-\tilde{\lambda}_0(A))\tilde{\lambda}_d(A)\kappa(A_l^{(d)})\kappa(A_r^{(d)})$ with $d = \hat{d}_{j,k}$ and $A = \hat{A}_{j,k}$; otherwise, the partition sets in level $j+1$ are $\hat{A}_{j+1,2k} = A_l^{(\hat{d}_{j,k})}$ and $\hat{A}_{j+1,2k+1} = A_r^{(\hat{d}_{j,k})}$. Recall that all descendants of a pruned node are pruned by design; hence, for each $\hat{A}_{j,k}$ that is pruned, we randomly select a direction $d_0 \in D(\hat{A}_{j,k})$ and set $\hat{A}_{j+1,2k} = A_l^{(d_0)}$ and $\hat{A}_{j+1,2k+1} = A_r^{(d_0)}$. Note that the reconstructed image is invariant to this random choice as all pixels in the pruned block will be reconstructed as a common constant.

3.6. Encoder/decoder and compressed structures

Given the permutation of the original image induced by $\hat{\mathcal{T}}$, under which the order of each pixel is given by binary coding of the branch under $\mathcal{T}$ to which each pixel belongs, we have a vectorization of the original image. Within a pruned node, the ordering of the pixels is arbitrary. At this point, one has the flexibility of choosing the favorite encoder and decoder for this vectorized image. In addition, the encoding part in Figure 2 includes a symbol encoder, which is used to reduce the coding redundancy. In particular, we implement two encoding/decoding algorithms in CARP depending on the image resolution to strike a balance between compression efficiency and runtime. For target images with high resolution, we adopt the Huffman encoding method based on the Huffman table that is derived from the estimated probability or frequency of occurrence for each possible value of the source symbol. The reduced symbols are stored as the compressed representation. This Huffman table is also used in the decoder part to perform the inverse operation of the symbol encoder. Arithmetic encoding can be used to further optimize the compression performance but may require longer execution time [20]. As such, we implement Huffman coding for

images with resolution 512×512 or higher and arithmetic coding otherwise. We find this improves the performance of CARP for images/videos with low resolution compared to our initial implementation in the conference version [8] (see Figure S4 in the Supplementary Material). Our code allows users to choose between these two encoding/decoding methods for images with any resolution.

In our following numerical experiments, we use the 1D Haar DWT and a symbol encoder as the encoder part while a symbol decoder and the inverse DWT as the decoder part, respectively, due to their computational scalability.

3.7. Empirical Bayes for setting hyperparameters

We specify the two mappings $\rho(\cdot)$ and $\eta(\cdot)$ as well as parameters τ_j by reparameterizing them using five hyperparameters $(\alpha, \beta, C, \tau_0, \eta_0)$: $\rho(A) = \min(1, C\, 2^{-\beta j})$ for any node A at the jth level, $\tau_j = 2^{-\alpha j}\tau_0$, and $\eta(A) = \eta_0$ for any node $A \in \mathcal{A}$. We use an empirical Bayes strategy to set the hyperparameters by maximizing the marginal likelihood $\Psi(\Omega)$ over a grid.

We observe that specifying the hyperparameters at fixed values eliminates the need for computing the maximum likelihood estimates without sacrificing compression efficiency much. As such, our software allows both options. Under either option, a user just needs to specify a single parameter σ to obtain images at desired compression ratios when applying CARP.

4. Experiments

In this section, we compare CARP with a wide range of popular compression methods using a variety of benchmark databases. In particular, we use a 2D still image dataset from the 2020 CLIC workshop and challenge `http://challenge.compression.cc/tasks`, a YouTube video dataset from [21], and a surveillance video dataset from [22]; see the Supplementary Material (Figure S1) for selected examples of all the datasets.

CARP and its software implementation are readily applicable to all these types of images, while the competitors may tailor to images of a particular dimension. We thus compare CARP with a different batch of methods depending

on the image type. In this section, we opt for fixed hyperparameters for simplicity. In particular, we use $\alpha = 0.5, \beta = 1, C = 0.05, \tau_0 = 1/\sigma$, and $\eta_0 = 0.4$. For RGB images or videos, we compress each RGB channel separately when implementing CARP to account for unique channel-specific structures and also in view of its simplicity for implementation.

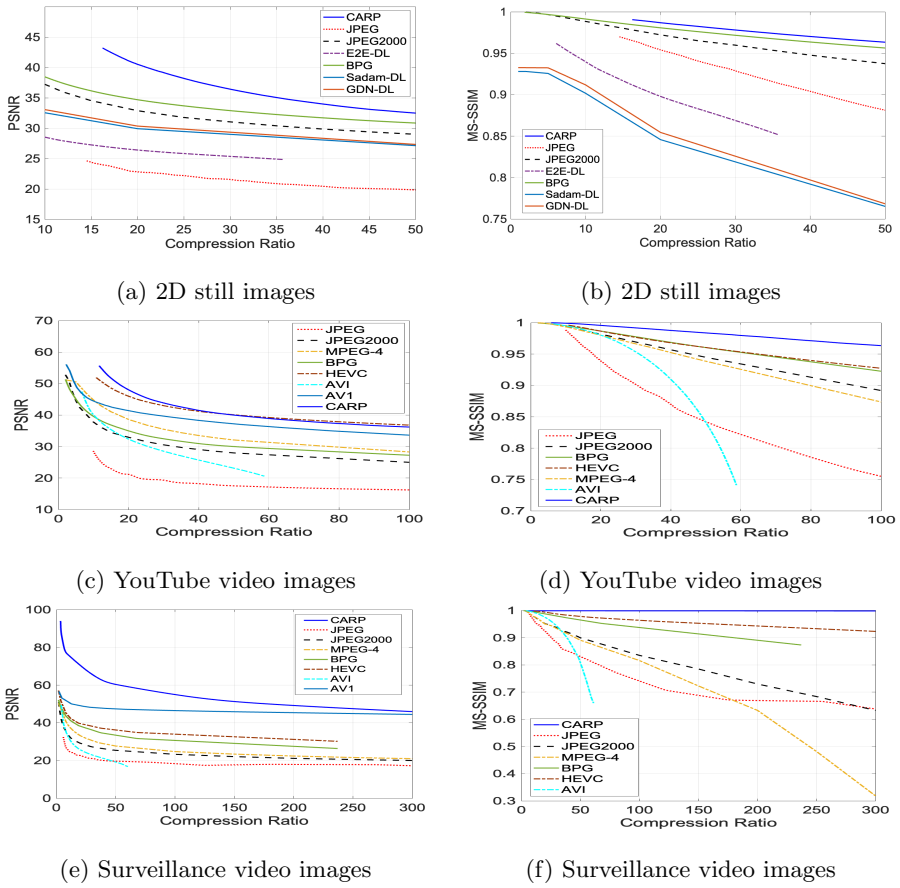
4.1. Average performance

Figure 3 summarizes the average performance on three image/video types under the metric of peak signal-to-noise ratio (PSNR) and multi-scale structural similarity (MS-SSIM), the latter being a widely used metric to assess perceived image quality and measure the structural similarity between two images; in all of our test data CARP compares favorably to a variety of competitors—including JPEG, JPEG2000, AVI, End-to-End deep learning (E2E-DL), Sadam deep learning (Sadam-DL), GDN deep learning (GDN-DL), BPG, MPEG4, HEVC, AV1 under both PSNR and MS-SSIM, and appears to give the leading performance on average in most cases. We note that in Figure 3(a) and 3(b), the averages of metrics are calculated over a subset of 70 images from the image dataset on which the methods being compared are able to achieve a wide range of compression ratios. Also, we acknowledge that because we have used three pre-trained deep learning models[5, 10], the performance of these methods could be improved had the CNNs been trained on images that are particularly suited for the 2D still image database. In any case, we note the robust performance of CARP over all the tests and the fact that it does not require separate pre-training on additional images.

Next we present more detailed numerical results that compare the image-specific performance of the methods, which show that CARP can outperform the competitors in nearly all of the individual images in some datasets we have examined.

4.2. 2D still images

We compare CARP with JPEG, JPEG2000, BPG[3], and three pre-trained deep learning methods including 'E2E-DL'[5] and 'Sadam-DL' and 'GDN-DL'



(a) 2D still images

(b) 2D still images

(c) YouTube video images

(d) YouTube video images

(e) Surveillance video images

(f) Surveillance video images

Figure 3: Performance summary of CARP and competitors in three databases consisting of still images and videos at different resolutions. Each plot of the first column presents the average of peak-signal-to-noise-ratio (PSNR) while each plot of the second column presents the average of MS-SSIM at various compression ratios.

[10]. Here we randomly select 100 images from the 2020 CLIC workshop and challenge, which are resized to 512×512 to test each method. These 100 images are provided in the GitHub repository.

To assess each method, we use the peak signal-to-noise ratio (PSNR) and the multi-scale structural similarity index (MS-SSIM) of reconstructed images

at various compression ratios, which is further supplemented by visual comparison. Specifically, at various compression ratios, each 2D image is compressed and reconstructed, then the PSNR and MS-SSIM are calculated using the reconstructed image. Figure 3(a) shows CARP gives the best average PSNR while Figure 3(b) shows CARP gives the best average MS-SSIM at all compression ratios. In Figure 4, the first row plots the PSNR (in (a)) and MS-SSIM curves (in (b)) for CARP for all 100 images, while other rows present the PSNR and MS-SSIM ratio curve between each alternative method and CARP for all 100 individual images—with values under 1 indicating CARP outperforms the competitor. The performances of all MS-SSIM ratio curves are consistent with those in PSNR ratio curves. CARP appears to outperform the six competitors for nearly all individual images and at all compression ratios up to 300 at which we are able to apply the competitor, except on a handful of images for JPEG2000 at very low compression ratios, Sadam-DL and GDN-DL at very high compression ratios and a couple of images for BPG. For this database, E2E-DL underperforms CARP substantially, but we acknowledge that part of the substantial performance gap could be narrowed had the CNNs been trained on images that are particularly suited for the specific database. Like JPEG and JPEG2000, CARP does not require external pre-training. In addition, the user does not need to specify any tuning parameter other than σ, which is equivalent to specifying the compression ratio.

The locally adaptive nature of CARP enhances its ability to preserve local details in the images. As an illustration, we visualize reconstructions with a particular focus on detailed features in an image using three selected images in Figure 5. The region of interest is marked in the original image, and we present zoom-in views of the region with a yellow and red block in the reconstructed images from various methods at one specific compression ratio. Overall, CARP, BPG, Sadam-DL, and GDN-DL clearly outperform the other three methods (JPEG, JPEG2000, and E2E-DL). CARP tends to preserve substantially more details in the reconstruction relative to other methods. We next take BGP as an example, and similar observations can be made when compared to other

methods. For the redbrick wall in the top of Figure 5, the reconstructed image by CARP appears sharper and warmer than BPG; for the building in the middle of Figure 5, the stairs in the yellow and red block by CARP are much clearer than BPG, and a further zoom-in into the stairs shows that CARP preserves more details; for the landscape image in the bottom of Figure 5, there are more recognizable details in the mountain in sub-figure (b) but more blurry in sub-figure (c).

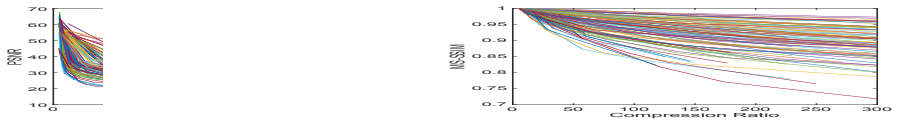
These observations may be partially explained by the design of CARP. The use of RDP in CARP cuts the image horizontally or vertically, which leads to its superiority when the target image has repeated patterns along the horizontal or vertical direction, e.g., the building in Img5 (the middle two rows of Figure 5). The pruning option in CARP increases the efficiency of image compression, which helps detect the blocks in an image with similar intensities and efficiently convert the tiles into pixel vectors, e.g., patches of the large dark bottom region in Img9 (the last two rows of Figure 5).

4.3. YouTube video dataset

We use the YouTube dataset in [21], which consists of instructional videos for five different tasks, including making a coffee, changing a car tire, performing cardiopulmonary resuscitation (CPR), jumping a car and re-potting a plant. The dataset has 150 videos with an average length of about 4,000 frames (or 2 minutes). Here we randomly select 20 videos from each task totaling 100 videos. Selected frames of the sampled videos are displayed in the Supplementary Material (Figure S1 and S2). Note that these YouTube videos have a low resolution of 256 by 256, which favor the MPEG-4 standard as MPEG-4 is optimized at low bit-rate video communications [23], and CARP gives competitive performance building on Arithmetic coding.

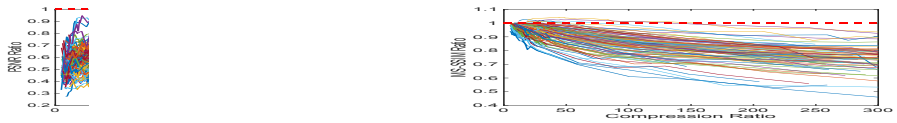
Video image data is produced through the compilation of 2D static images, which are the so-called "frames", synchronized at a given frame rate. When these frames are layered sequentially, the resulting video image transforms into a 3D representation, taking the time dimension into account. CARP is applicable

for streaming data by taking the entire video as input, thus constituting a genuine video compression method like MPEG-4. In addition to popular video compression standards, we also consider using JPEG and JPEG2000 through a frame-by-frame implementation, while CARP is directly applicable to 3D images with no modification. For HEVC, we use the settings of x265 and P-frame. For AV1, the reconstructed images do not allow direct access to the raw pixel intensities without additional output conversion but it reports the corresponding PSNR and SSIM (but not MS-SSIM) directly.



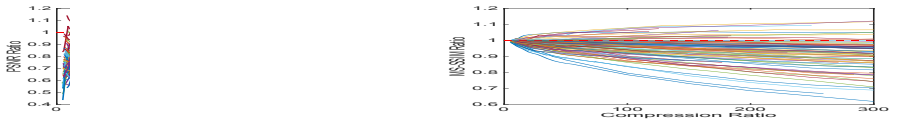
(a) PSNRs of CARP for 100 images

(b) MS-SSIMs of CARP for 100 images



(c) PSNR ratio: JPEG/CARP

(d) MS-SSIM ratio: JPEG/CARP



(e) PSNR ratio: JPEG2000/CARP

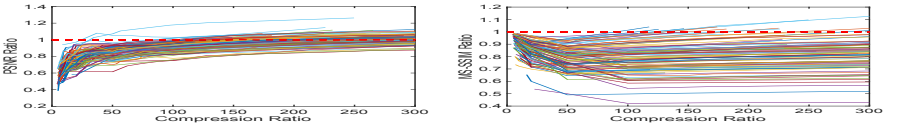
(f) MS-SSIM ratio: JPEG2000/CARP

(g) PSNR ratio: E2E-DL/CARP

(h) MS-SSIM ratio: E2E-DL/CARP

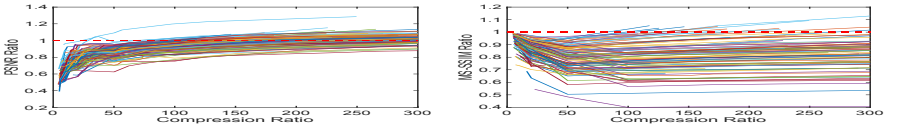
(i) PSNR ratio: BPG/CARP

(j) MS-SSIM ratio: BPG/CARP



(k) PSNR ratio: Sadam-DL/CARP

(l) MS-SSIM ratio: Sadam-DL/CARP



(m) PSNR ratio: GDN-DL/CARP

(n) MS-SSIM ratio: GDN-DL/CARP

Figure 4: 2D still images: PSNRs of CARP for 100 individual images in (a) and MS-SSIM of CARP for 100 individual images in (b); PSNR ratio curves and MS-SSIM ratio curves for JPEG, JPEG2000, E2E-DL, BPG, Sadam-DL and GDN-DL relative to CARP in (c)- (n), respectively.

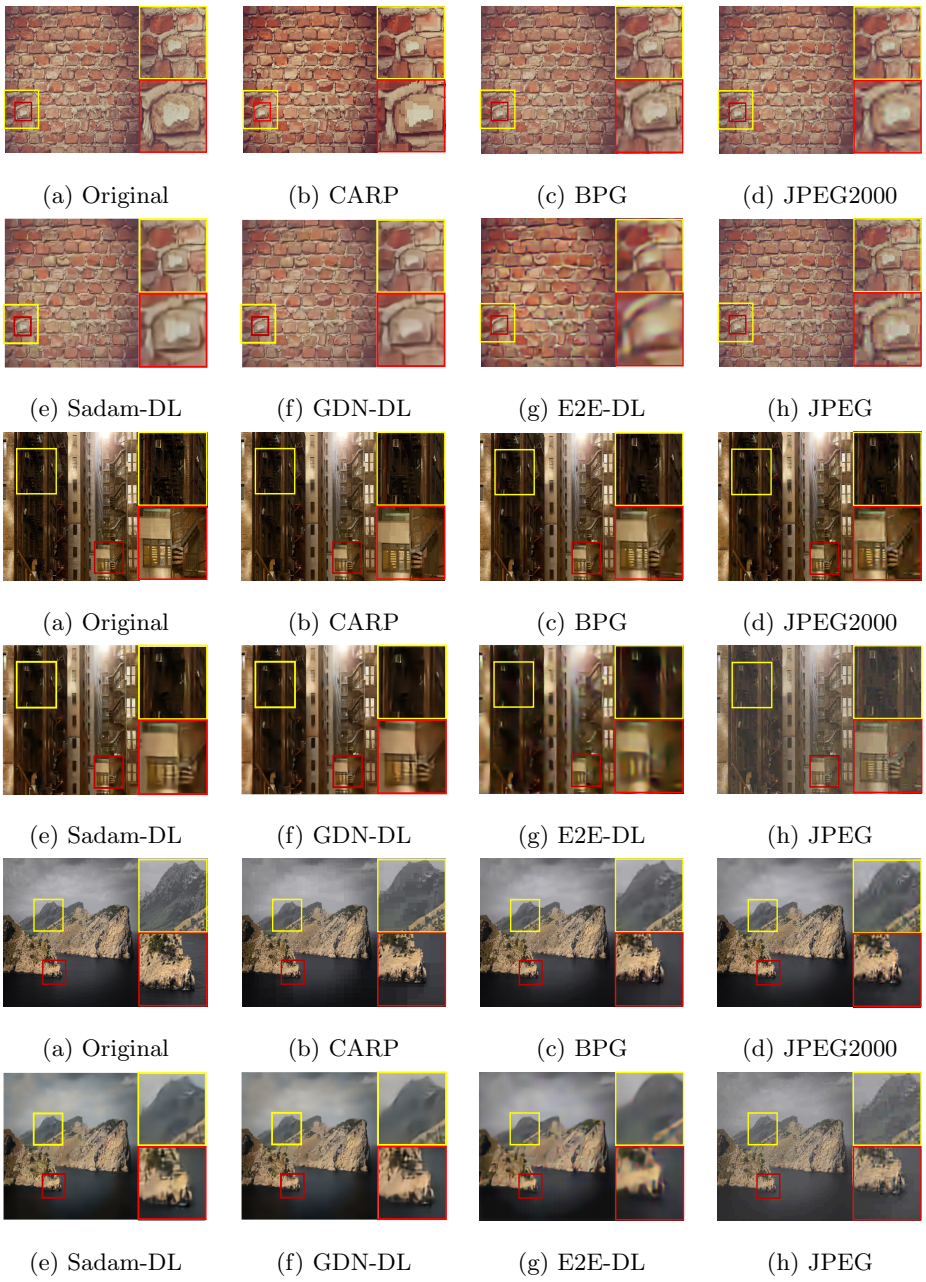


Figure 5: Comparison of reconstructed images. The compression ratio is 45 for Img1 (top), 66 for Img5 (middle), and 45 for Img9 (bottom).

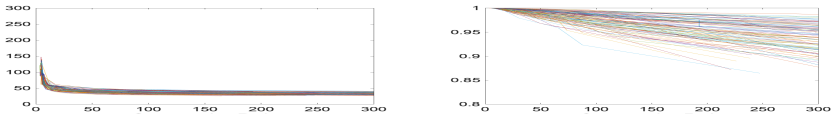
Figure 6 presents the PSNR and MS-SSIM ratio curves (alternative methods over CARP) at various compression ratios for all the 100 videos, as well as the PSNR and MS-SSIM curves of CARP. CARP substantially outperforms JPEG, JPEG2000, AVI, and BPG for nearly all individual videos at all compression ratios. CARP gives larger PNSR than MPEG-4; for compression ratios below 150, MPEG-4 and CARP perform similarly at a subset of the videos in MS-SSIM; for compression ratios above 150, CARP increasingly outperforms MPEG-4 at all videos. We note again that all the videos are at a low resolution that substantially favors MPEG-4.

HEVC leads to larger PSNR than CARP on most videos when the compression ratio is larger than 10 as shown in Figure 6(k), while CARP outperforms HEVC in MS-SSIM on almost all the individual videos as shown in Figure 6(l). Overall, Figure 3 shows that CARP tends to be the leading approach on average under the MS-SSIM metric, and is among the best methods in PSNR where it underperforms HEVC by a small margin in a subset of samples. See the Supplementary Material for the performance comparison between CARP with Huffman and CARP with Arithmetic encoder/decoder (Figure S4).

For visual comparison, we select a video from the "replotting a plant" task and compare one frame of the reconstructed video to that of the original one in Figure 7. The zoomed region shown in the bottom row shows that the reconstructed frame via CARP captures most details in the original frame (e.g., the words on the label, particularly the letters in front of "V", and the white dots below the label). The reconstructed image by AV1 is clearer and smoother than other images, but it relatively deviates more from the original image.

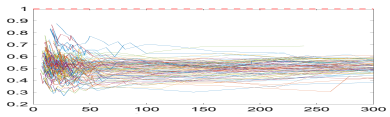
4.4. Surveillance video dataset

We next investigate the performance of CARP on higher-resolution videos through a surveillance video dataset [22], where each video has a resolution of 1024 by 1024. We randomly select one surveillance video for a parking lot.

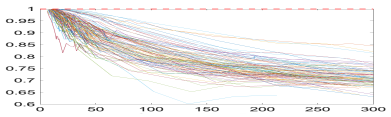


(a) PSNR of CARP for 100 videos

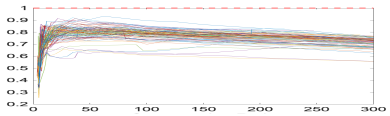
(b) MS-SSIM of CARP for 100 videos



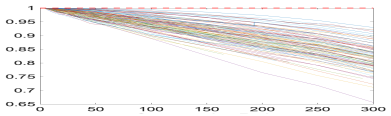
(c) PSNR ratio: JPEG/CARP



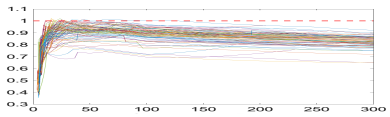
(d) MS-SSIM ratio: JPEG/CARP



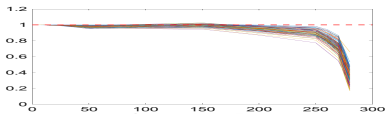
(e) PSNR ratio: JPEG2000/CARP



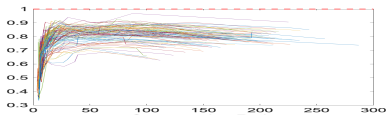
(f) MS-SSIM ratio: JPEG2000/CARP



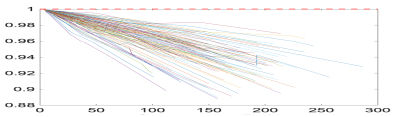
(g) PSNR ratio: MPEG-4/CARP



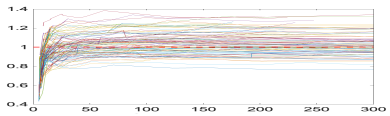
(h) MS-SSIM ratio: MPEG-4/CARP



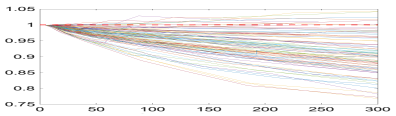
(i) PSNR ratio: BPG/CARP



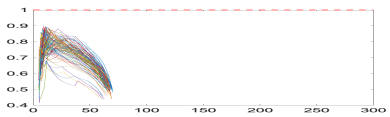
(j) MS-SSIM ratio: BPG/CARP



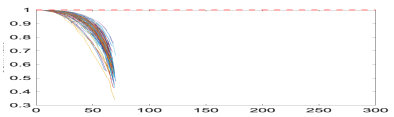
(k) PSNR ratio: HEVC/CARP



(l) MS-SSIM ratio: HEVC/CARP



(m) PSNR ratio: AVI/CARP



(n) MS-SSIM ratio: AVI/CARP

Figure 6: YouTube videos: PSNR curves and MS-SSIM curves of CARP for 100 videos in (a) and (b); PSNR ratio curve and MS-SSIM ratio curves of JPEG, JPEG200, MPEG-4, BPG, HEVC and AVI relative to CARP in (c)–(n), respectively.

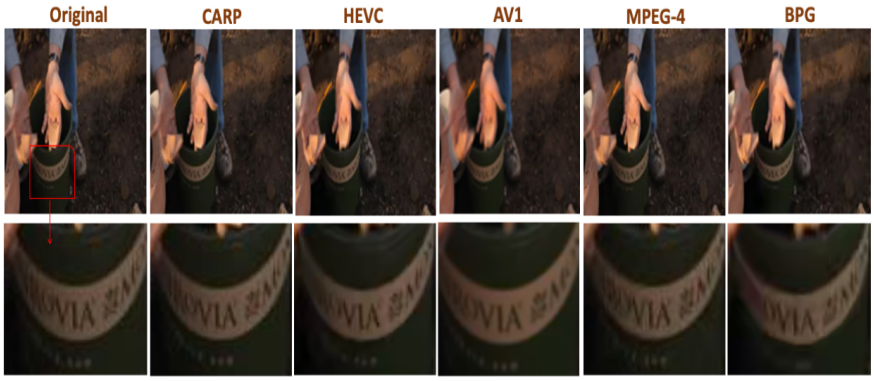


Figure 7: Selected frame of reconstructed YouTube videos by CARP, HEVC, AV1, MPEG-4, and BPG at compression ratio 30. See the Supplementary Material for the reconstruction by JPEG2000, JPEG, and AVI (Figure S5), and reconstructed continuous frames by CARP (Figure S3).

We divide the entire video into 180 segments of equal length to help assess the longitudinal variability of compression performances of each method and reduce the computational time of each method. Figure 8 plots the PSNR and MS-SSIM ratio curves (alternative method over CARP) among all the 180 videos as well as the PSNR and MS-SSIM curve for CARP at various compression ratios. We can see that CARP gives the best PSNRs and MS-SSIMs for almost all videos at all compression ratios (up to 300), with only one exception in Figure 8(k) where CARP gives slightly smaller PSNR than HEVC on one video when the compression ratio is around 250-300. Overall, CARP gives better average PSNR and MS-SSIM than all other methods at all compression ratios from 0 to 300, as shown in Figure 3(e) and Figure 3(f). Comparing the two video datasets (second and third rows in Figure 3) suggests that although its excellent performance is robust across datasets, CARP appears particularly well suited for the surveillance video dataset, which is expected as the relatively similar image patterns and backgrounds across frames are more compressible through partitions and pruning.

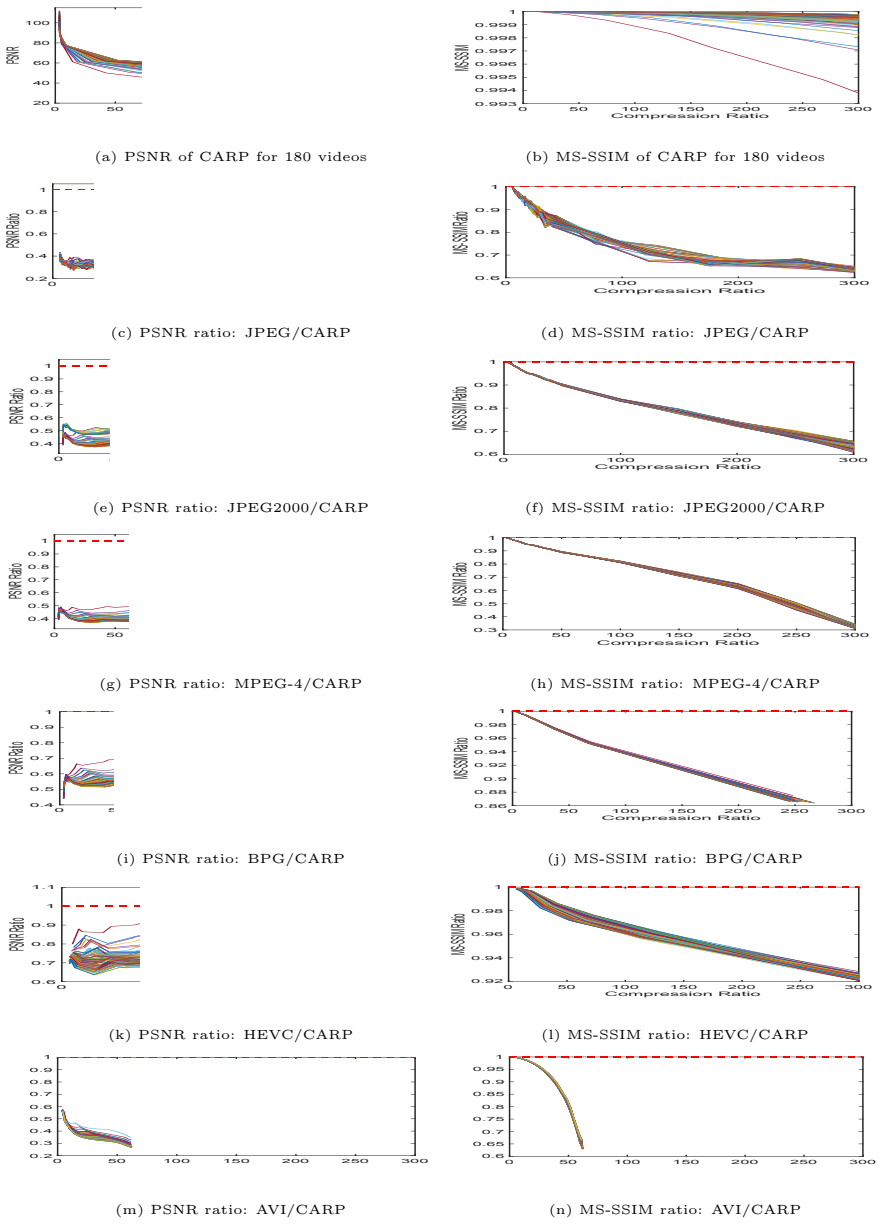
For visual comparison, we randomly select one video and compare one frame

of reconstructed videos. Figure 9 shows the original frame and reconstructed frames by CARP, AV1, HEVC, BPG, and MPEG-4, when the compression ratio is set to 30. The zoomed region is the shadow area at the top-right corner in the original frame, shown in the bottom row. In comparison, the reconstructed frame via CARP captures most details of the region in the original frame, while the region reconstructed via MPEG-4 is more blurry (e.g., the edge of the yellow arrow). The arrows and their background are smoother in HEVC, AV1, and BPG than in CARP, but the reconstructed objects by CARP seem closer to the original pattern.

5. Discussion

CARP uses a principled Bayesian hierarchical model to learn a data-adaptive permutation on the image space, which allows effective wavelet transforms on the image/video, achieving in-situ compression along with self-tuning and progressive transmission. We conduct extensive experiments and show that CARP compares favorably to a wide range of popular image/video compression methods for a variety of image types.

CARP is computationally efficient in that it scales linearly with the number of pixels of an image. Taking the 2D still image database as an example, the average encoding time under our implementation of CARP, without any parallel computing, is around 3.17 second/image, compared to 0.82 second/image for JPEG, 0.40 second/image for JPEG2000, and 88.75 second/image for E2E-DL; the average decoding time for CARP is around 0.91 second/image, compared to 2.30 second/image for JPEG, 0.03 second/image for JPEG2000, and 3.62 second/image for E2E-DL, tested on a Macbook Pro with 2.2 GHz Intel Core i7 processor. The computing time of CARP can be further reduced with more optimized implementation. In particular, one main computational task in CARP is to compute the marginal likelihood of the wavelet regression model on each node in the partition tree, which can be parallelized over the nodes in the partition tree.



(a) PSNR of CARP for 180 videos

(b) MS-SSIM of CARP for 180 videos

(c) PSNR ratio: JPEG/CARP

(d) MS-SSIM ratio: JPEG/CARP

(e) PSNR ratio: JPEG2000/CARP

(f) MS-SSIM ratio: JPEG2000/CARP

(g) PSNR ratio: MPEG-4/CARP

(h) MS-SSIM ratio: MPEG-4/CARP

(i) PSNR ratio: BPG/CARP

(j) MS-SSIM ratio: BPG/CARP

(k) PSNR ratio: HEVC/CARP

(l) MS-SSIM ratio: HEVC/CARP

(m) PSNR ratio: AVI/CARP

(n) MS-SSIM ratio: AVI/CARP

Figure 8: Surveillance videos: PSNR curves and MS-SSIM curves of CARP for 180 videos in (a) and (b); PSNR ratio curve and MS-SSIM ratio curves of JPEG, JPEG200, MPEG-4, BPG, HEVC and AVI relative to CARP in (c)–(n), respectively.

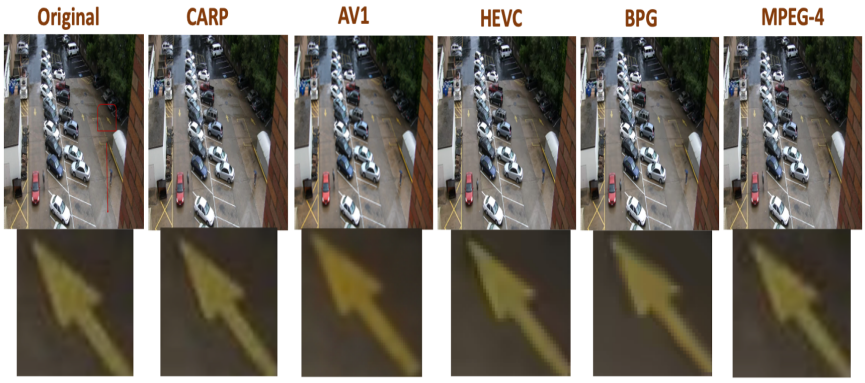


Figure 9: Selected frame of reconstructed surveillance videos by five methods when compression ratio is set to 30. Left to right: Original frame, CARP, AV1, HEVC, BPG, MPEG-4. See the Supplementary Material (Figure S6) for the reconstruction by JPEG2000, JPEG, and AVI.

There is a limitation associated with the implementation of our CARP approach. It imposes a very strict requirement on the input size, necessitating image preprocessing by resizing each dimension to 2^n if they do not already meet this specification. There are also several other interesting future directions building on our CARP approach. First, CARP complements a wide range of encoding/decoding methods for 1D vectors. Instead of Huffman coding, we can use other entropy encoding methods such as arithmetic coding or adaptive methods on 1D vectors to further improve the compression ratio. Second, it is possible to adopt a different point estimate within the CARP framework for improved compression. In particular, one may consider another posterior summary method that leads to reconstruction beyond MAP. Finally, for RGB images, although CARP is able to treat the channels together as a 3D image, there is room to further improve this strategy. For example, we can learn the partition based on one channel and share it with other channels since the image patterns in the three channels may be similar to each other.

CARP provides a flexible framework to enable application-specific adaptation. One example is to achieve low latency in real-time streaming video

compression. In particular, one may utilize progressive transmission enabled by CARP to gradually improve the image quality as data bits are incrementally transmitted, or simply process videos by smaller segments, where the buffering is reduced as little temporary storage will be taken by the uncompressed or compressed video data. It is also interesting to develop an online version of CARP to transmit the tree structure of each frame or each sub-video dynamically to improve latency, but one needs to properly account for possible frame-to-frame inconsistency and error propagation.

Another example is visual enhancement in compressed images. The visual comparison in Figure 5 shows sharp boundaries at high compression ratios as the tree-based multi-scale representation in CARP uses a node for a sub-block of the image, and blockwise shrinkage is adopted via pruning. While this feature is key to preserving local details compared to other approaches, the most desirable balance between contrast and smoothness may vary in different applications. If desired in certain applications, there are several strategies to mitigate the blockiness. For example, one may vary the prior pruning probabilities to achieve a different level of smoothing in the visualization. Alternatively, we can add post-processing modules via intensity smoothing to reduce the contrast of adjacent pixels. In addition, the cycle spinning technique has been used to enhance visualization of multi-scale methods in image/video reconstruction, and an adapted strategy may be considered for compression by focusing on finer scales of the image/video to strike a balance between reconstruction quality and model parsimony.

References

[1] G. K. Wallace, The JPEG still picture compression standard, IEEE Transactions on Consumer Electronics 38 (1) (1992) xviii–xxxiv.

[2] A. Skodras, C. Christopoulos, T. Ebrahimi, The JPEG 2000 still image compression standard, IEEE Signal Processing Magazine 18 (5) (2001) 36–58.

[3] F. Bellard, BPG image format, `http://bellard.org/bpg/` (accessed: 2018-04-21).

[4] H. Liu, T. Chen, Q. Shen, T. Yue, Z. Ma, Deep image compression via end-to-end learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 2575–2578.

[5] J. Ballé, V. Laparra, E. P. Simoncelli, End-to-end Optimized Image Compression, in: International Conference on Learning Representations, 2017.

[6] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, L. V. Gool, Soft-to-hard vector quantization for end-to-end learning compressible representations, in: Advances in Neural Information Processing Systems, 2017, pp. 1141–1151.

[7] M. Marcellin, M. Gormish, A. Bilgin, M. Boliek, An overview of JPEG-2000, in: Proceedings DCC 2000. Data Compression Conference, IEEE, 2000, pp. 523–541.

[8] R. Liu, M. Li, L. Ma, CARP: Compression through adaptive recursive partitioning for multi-dimensional images, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14306–14314.

[9] M. Li, W. Zuo, S. Gu, D. Zhao, D. Zhang, Learning convolutional networks for content-weighted image compression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3214–3223.

[10] J. Ballé, Efficient nonlinear transforms for lossy image compression, in: 2018 Picture Coding Symposium (PCS), IEEE, 2018, pp. 248–252.

[11] J. Watkinson, The MPEG Handbook, Routledge, 2012.

[12] J. Chen, U.-V. Koc, R. Liu, Design of digital video coding systems: A complete compressed domain approach, CRC Press, 2001.

[13] G. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1649–1668.

[14] S. Anila, N. Devarajan, The usage of peak transform for image compression, International Journal of Computer Systems Science & Engineering 2 (11) (2010) 6308–6316.

[15] M. Li, L. Ma, Learning asymmetric and local features in multi-dimensional data through wavelets with recursive partitioning, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (11) (2022) 7674–7687.

[16] T. Chujoh, S. Koto, Y. Kikuchi, Video encoding/decoding method and apparatus, US Patent 7,680,184 (Mar. 16 2010).

[17] H. A. Chipman, E. D. Kolaczyk, R. E. McCulloch, Adaptive Bayesian wavelet shrinkage, Journal of the American Statistical Association 92 (440) (1997) 1413–1421.

[18] M. Clyde, E. George, Flexible empirical Bayes estimation for wavelets, Journal of the Royal Statistical Society: Series B 62 (4) (2000) 681–698.

[19] M. Crouse, R. Nowak, R. Baraniuk, Wavelet-based statistical signal processing using hidden Markov models, IEEE Transactions on Signal Processing 46 (4) (1998) 886–902.

[20] S. Kaur, M. Sukhjeet, Entropy coding and different coding techniques, Journal of Network Communication and Emerging Technologies 6 (2016) 4–7.

[21] J.-B. Alayrac, P. Bojanowski, N. Agrawal, J. Sivic, I. Laptev, S. Lacoste-Julien, Unsupervised learning from narrated instruction videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4575–4583.

[22] R. Vezzani, R. Cucchiara, Video surveillance online repository (visor): an integrated framework, Multimedia Tools and Applications 50 (2) (2010) 359–380.

[23] T. Sikora, MPEG-4 very low bit rate video, in: 1997 IEEE International Symposium on Circuits and Systems (ISCAS), Vol. 2, IEEE, 1997, pp. 1440–1443.