This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

CARP: Compression through Adaptive Recursive Partitioning for Multi-dimensional Images

Rongjie Liu
Rice University
rongjie.liu@rice.edu

Meng Li
Rice University
meng@rice.edu

Li Ma
Duke University
li.ma@duke.edu

Abstract

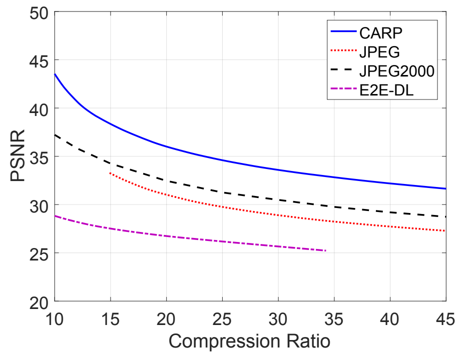
Fast and effective image compression for multi-dimensional images has become increasingly important for efficient storage and transfer of massive amounts of high-resolution images and videos. Desirable properties in compression methods include (1) high reconstruction quality at a wide range of compression rates while preserving key local details, (2) computational scalability, (3) applicability to a variety of different image/video types and of different dimensions, and (4) ease of tuning. We present such a method for multi-dimensional image 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. CARP uses a multi-layer Bayesian hierarchical model to achieve self-tuning and regularization to avoid overfitting—resulting in one single parameter to be specified by the user to achieve the desired compression rate. Extensive numerical experiments using a variety of datasets including 2D ImageNet, 3D medical image, and real-life YouTube and surveillance videos show that CARP dominates the state-of-the-art compression approaches—including JPEG, JPEG2000, MPEG4, and a neural network-based method—for all of these different image types and often on nearly all of the individual images.

1. Introduction

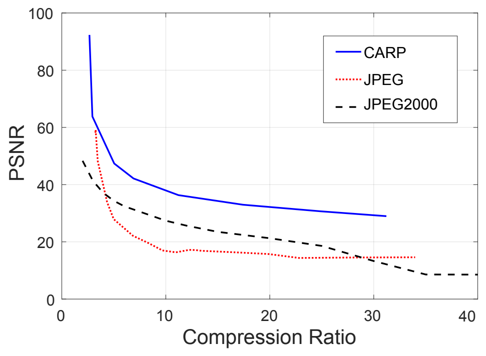
Image compression is a long-standing, fundamental problem in computer vision and image processing, and is key to efficient storage and transfer of the vast amount of high-resolution images and videos that are routinely collected in a variety of applications. Efficient compression relies on parsimonious representations of images that preserve important spatial and contextual features. Standards such as JPEG and JPEG2000 utilize fixed, deterministic linear function transforms, such as wavelets, followed by optimized encoding under such transforms. These approaches give excellent stability and scalability in practical implementation, and require little training and tuning. However, they lack adaptivity to image-specific features and as such achieve only suboptimal compression efficiency. The more recent convolutional neural network (CNN) based approaches utilize much more flexible, nonlinear transformations of the original image. This additional flexibility often leads to improved compression efficiency, but at the same time leads to substantially more extensive training and tuning of the methods.

We aim to strike a middle ground between these two approaches by introducing a Bayesian probabilistic modeling strategy for incorporating adaptivity to image features into the wavelet transform based image processing framework, while maintaining its computational scalability and ease of tuning. Instead of using fixed wavelet transforms, we treat the transform as an unknown latent quantity and learn an optimal transform by placing a Bayesian prior on the space of such transforms induced by random recursive partitioning on the image and compute the *maximum a posteriori* (MAP) estimate of the transform under our model. A compressed image can then be produced under the inferred transform that tailors to image-specific features. Moreover, such computation is as efficient as the classical wavelet-based methods—scaling linearly with the size of the image.

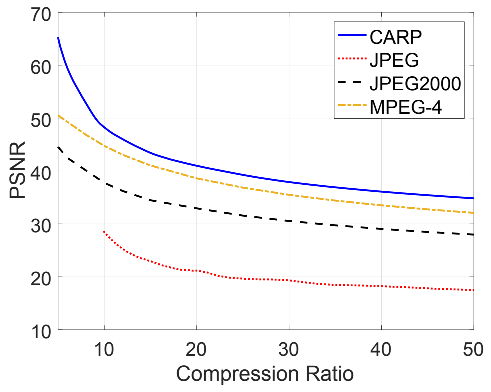
Aside from achieving excellent compression efficiency (to be demonstrated in our numerical experiments), our method, called Compression via Adaptive Recursive Partitioning (CARP), enjoys two additional advantages. First, it is directly applicable to images of different dimensions without modification, 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



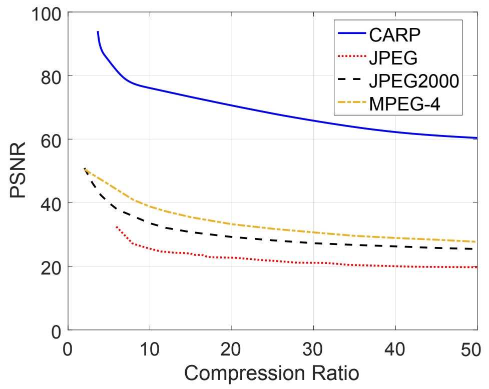
(a) 2D ImageNet images



(b) 3D brain image



(c) YouTube video dataset



(d) Surveillance video dataset

Figure 1. Performance summary of CARP and competitors in four databases consisting of still images (2D and 3D) and videos. Each plot presents the peak-signal-to-noise-ratio (PSNR) at various compression ratios for each method. Details are given in Section 4.

to use, especially by common users, without requiring expert knowledge of the underlying method.

To validate the efficiency and robustness of our method, we use a variety of benchmark databases to compare CARP with several state-of-the-art compression methods for 2D and 3D images. Figure 1 summarizes the average performance on four image/video types, and in all of them CARP dominates the state-of-the-art competitors—including JPEG, JPEG2000, End-to-End deep learning (E2E-DL) and MPEG4. We note that in Figure 1(a) the average PSNRs are calculated over a subset of 70 images from the ImageNet on which the methods being compared are able to achieve a wide range of compression ratios (from 15 to 35). Also, we have used a pre-trained model for E2E-DL [5], and so part of the substantial performance gap between E2E-DL from other methods could be narrowed had the CNNs been trained on images that are particularly suited for the ImageNet database. The code for average performance is also available on Github for reproducibility. In Section 4 we present more detailed numerical results that compare the image-specific performance of the methods, which show that CARP in fact often dominates the competitors in nearly all of the individual images we have examined.

2. Related work

For 2D images, perhaps the most well-known image compression algorithms are JPEG [23] and its successor JPEG2000 [21]. 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 multiscale orthogonal wavelet decomposition with arithmetic coding. However, both JPEG and JPEG2000 are suboptimal for image compression [14] due to non-adaptive image transformation and a separate optimization on codecs.

Besides JPEG and JPEG2000, there is a growing literature in developing deep learning-based methods [15, 5, 1, 14, 4, 18] 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, 14]. 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.

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 and JPEG2000), the critical temporal redundancy is undesirably ignored. Thus, most video compression algorithms in Moving Picture Experts Group (MPEG) [24] exploit both spatial and temporal redundancy. For example, MPEG-4 absorbs many features of different standards using both DCT and motion compensation [7] 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 be not globally optimal and hurdle a more efficient compression.

Recursive partitioning induces a permutation of the pixels and has been used previously in other applications. In particular, [3] adopts peak transform to obtain spatial permutation. [13] uses random recursive partitioning to induce a prior 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 wavelet transforms, but instead aim at learning an optimal transform to represent the image thereby achieving efficient compression.

3. Method

3.1. CARP: The framework

In this paper, we develop a framework for Compression via Adaptive Recursive Partitioning (CARP). CARP uses an optimal 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. Because the space of all permutations is massive and only those permutations that preserve spatial features in an image can provide efficient representations, CARP utilizes a Bayesian prior on the space of permutations induced by random recursive partitioning along a bifurcating tree. This random recursive partitioning incorporates a pruning option to probabilistic 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. In this work, we use the 1D discrete wavelet transform (DWT) and Huffman symbol encoding algorithm as the encoder, and use the inverse DWI and Huffman symbol decoding algorithm as the decoder.

Figure 2 presents the workflow of CARP, where the two black boxes pinpoint the key techniques used in CARP. We next describe details of the pipeline of CARP, annotated by a toy example given by the 4 by 4 image in Figure 2 (also plotted at the top of Figure 3) to ease demonstration.

Input CARP takes an m-dimensional image $\boldsymbol{y}$ observed at an m-dimensional rectangular "pixel" space Ω of the form

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

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 $J = \sum_{i=1}^{m} J_i$. In our toy example in Figure 3, we have $m = 2$, $J_1 = J_2 = 2$, and $J = 4$.

The effectiveness of the compression highly depends on the representation power of the transform in use, especially its adaptivity to local and spatial features in an image. In CARP, this is achieved by a Bayesian probabilistic modeling strategy, where adaptivity to image features is incorporated into a wavelet transform based multiscale 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

While multiscale 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.

Considering all $n!$ permutations of pixels in Ω is not only computationally prohibitive but wasteful as well because the vast majority of permutations ignores 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 $j = 0, \ldots, J$.

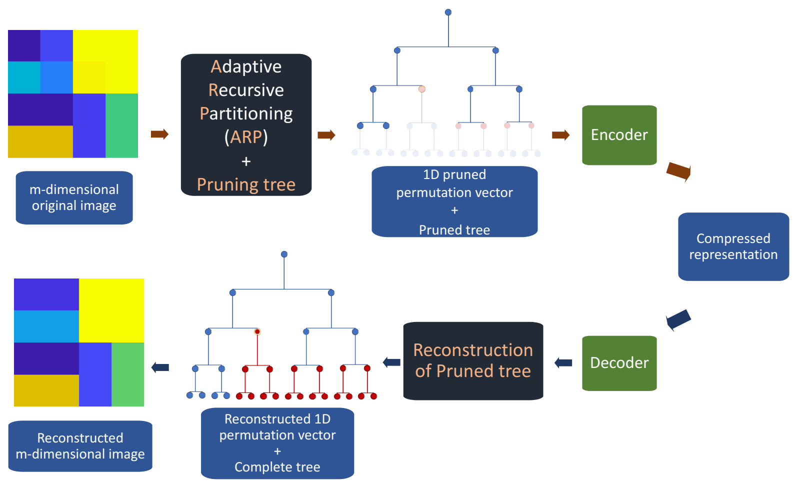

m-dimensional original image
Adaptive Recursive Partitioning (ARP) + Pruning tree
1D pruned permutation vector + Pruned tree
Encoder
Compressed representation
Decoder
Reconstruction of Pruned tree
Reconstructed 1D permutation vector + Complete tree
Reconstructed m-dimensional image

Figure 2. The workflow of CARP.

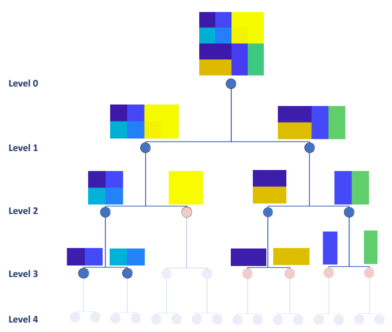


Figure 3. Toy example to demonstrate RDP and pruning: the partition tree is obtained by MAP.

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. Figure 3 shows one RDP from Level 0 (Ω) down to Level 4, where all pixels at Level 4 are ordered according to the induced permutation of Ω.

3.3. Bayesian modeling of RDPs

We use a hierarchical Bayesian model to adaptively learn an optimal RDP through maximizing the posterior distribution. To this end, we adopt a generative distribution called "random RDP" (RRDP) proposed in [25, 16, 13] as the prior on the RDP. For any node A in $\mathcal{T}$, RRDP specifies the probability of partitioning A in its ith dimension for $i = 1, \ldots, m$ using a vector-valued hyperparameter $\boldsymbol{\lambda}(A)$. We use $\boldsymbol{\lambda}$ to collect all these parameters and indicate with $\mathcal{T} \sim \text{RRDP}(\boldsymbol{\lambda})$ the distribution of RRDP. By default, we set the value of $\boldsymbol{\lambda}$ to be such that all divisible dimensions of each A have equal probability to be divided.

3.4. Pruning by Markov-tree wavelet regression

Each RDP turns the pixel space Ω into a vector of the same length as the number of pixels. To achieve a parsimonious image representation, the next component of our model aims at pruning the RDP tree in nodes where the pixel intensities are similar enough. To this end, we use wavelet shrinkage to achieve the desired pruning. 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 thresholding and shrinkage on the wavelet coefficients [8, 9, 6, 12, 19], and we shall use a Bayesian wavelet regression model with a Markov tree prior on the wavelet coefficients to achieve adaptive shrinkage [10]. 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

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

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

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 [10] 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 $(\rho_{j,k}(1-\eta_{j,k}), (1-\rho_{j,k})(1-\eta_{j,k}), \eta_{j,k})$, respectively. We assume $u_{j,k} \sim N(0, \sigma^2)$ independently across j and k. 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 ignores in the 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 will be the only parameter the user will set.

3.5. Posterior inference

For image compression, we need to find a single representative RDP that most effectively represent features in the image. 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 $\hat{\mathcal{T}}$.

To this end, we first need to find the marginal posterior distribution for $\mathcal{T}$. It turns out that under the above model, the posterior of $\mathcal{T}$ is conjugate—it is still an RRDP distribution, but with updated posterior selection probabilities, $\tilde{\boldsymbol{\lambda}}$, where we use tilde to indicate the posterior updated values for the parameters $\boldsymbol{\lambda}$. Theorem 1 of [13] provides an exact forward-backward algorithm for analytically computing the values for $\tilde{\boldsymbol{\lambda}}$ with computational complexity $O(n)$. Finally, based on the marginal posterior, we compute the MAP tree $\hat{\mathcal{T}}$ by standard bottom-up dynamic programming, which again incurs complexity $O(n)$.

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 of this vectorized image. In our following numerical experiment, 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. For the symbol encoder, we adopt the Huffman encoding method to reduce coding redundancy. Specifically, the Huffman table is derived from the estimated probability or frequency of occurrence for each possible value of the source symbol. Furthermore, the reduced symbols are stored as the compressed representation, and the Huffman table is also used in the decoder part to perform the inverse operation of symbol encoder.

3.7. Empirical Bayes for setting hyperparameters

We specify the parameters $\rho_{j,k}$, $\tau_{j,k}$, and $\eta_{j,k}$ by reparameterizing them using five hyperparameters $(\alpha, \beta, C, \tau_0, \eta_0)$: $\rho_{j,k} = \min(1, C2^{-\beta j})$, $\tau_{j,k} = 2^{-\alpha j}\tau_0$, and $\eta_{j,k} = \eta_0$. We use an empirical Bayes strategy to set the hyperparameters by maximizing the marginal likelihood over a grid. We observe that specifying the hyperparameters at fixed values eliminates the need of computing maximum likelihood 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 several state-of-the-art compression methods using a variety of benchmark databases, including still images (2D and 3D) and videos of low and high resolutions. In particular, we use 2D natural images from the ImageNet database [11], a 3D brain image [17], a YouTube video dataset from [2], and a surveillance video dataset from [22]. Specifically, CARP package is available at Github: https://github.com/xylimeng/CARP. 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 $\eta_0 = 0.4$.

4.1. Still images: 2D ImageNet Images

We compare CARP with popular compression methods designed for 2D images, including JPEG, JPEG2000, 2D wavelet transformation, and a deep learning method. For the deep learning method, we adopted a pre-trained end-to-end optimized image compression method ('E2E-DL') in [5]. We also include a compression method termed '1D-JPEG2000', where JPEG2000 is applied to a vectorized 1D vector column by column. While 1D-JPEG2000 is not a standard approach for image compression, its comparison with CARP quantifies the performance gain by using adaptive vectorization that serves as a central technique in our CARP method.

The Fall 2011 release of the ImageNet database [11] consists of 14,197,122 urls. We here randomly select 100 images of 512 by 512 to test each method and all these 100 images are provided in the Github. Some of the 2D images are presented in Figure 4.

To assess each method, we use the peak signal-to-noise ratio (PSNR) of the 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 is calculated using the reconstructed image. Figure 1(a) shows CARP gives the best average PSNR at all compression ratios. Figure 5 plots the PSNR ratio curve between each alternative method and CARP—with values under 1 indicating CARP outperforms the competitor—for all 100 individual images as well as the PNSR curve for CARP for all 100 images. CARP almost uniformly outperforms all of the five 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 JPEG and JPEG2000 at very low compression ratios. For this database, E2E-DL un-

Figure 4. A subset of 2D images from the ImageNet dataset.

derperforms 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 ImageNet database. Like JPEG and JPEG2000, CARP does not require external pre-training and is applicable for compressing any single image; moreover, 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 one selected image, the bird image, in Figure 6. The region of interest is marked in the original image, and we present zoom-in views of the interesting region in the reconstructed images from six different methods (compression ratio is set to 30). CARP, JPEG, and JPEG2000 clearly outperform the other three methods (wavelet, E2E-DL, and 1D-JPEG2000) overall. A further zoom-in into the head region of the bird shows that CARP preserves the most details among all methods.

4.2. Still image: 3D brain image

We use a 3D magnetic resonance imaging (MRI) dataset of size $64 \times 64 \times 64$ [17] accessed from MATLAB to illustrate CARP in 3D settings. The middle slice in each of the three directions (axial view, sagittal view, and coronal view) are presented in the first row of Figure 7.

CARP is directly applicable to 3D images with no modification, whereas JPEG and JPEG2000 are not directly applicable to 3D images. We adapt JPEG and JPEG2000 slice-by-slice to compress 3D volumes, where each slice of the 3D image is compressed and then combined to form a 3D compression. The PSNR curves in Figure 1(b) shows that CARP dominates the other two methods for all compression ratios (up to 30). This is expected as CARP is not only

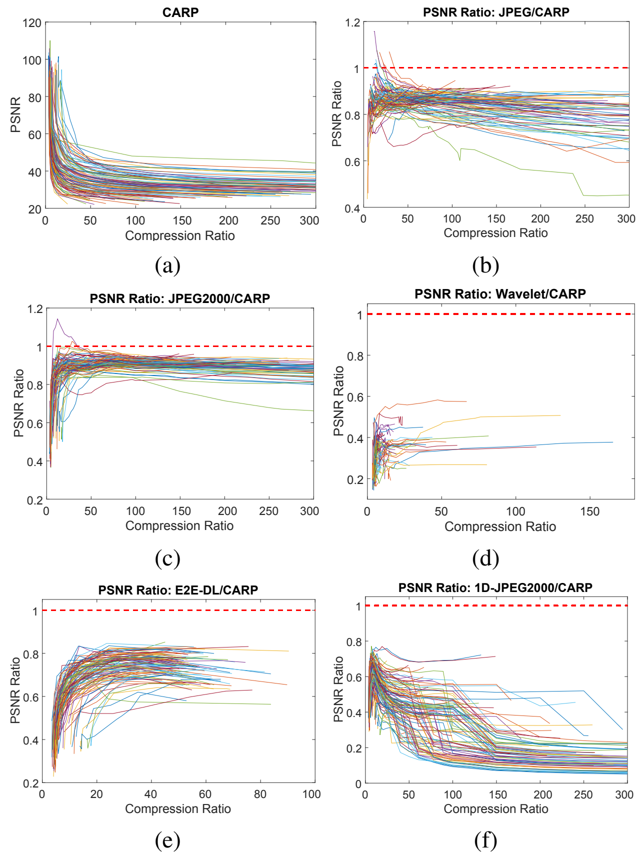


Figure 5. ImageNet images: PSNRs of CARP for 100 individual images in (a) and PSNR ratio curves for JPEG, JPEG2000, Wavelets, E2E-DL, 1D-JPEG2000 relative to CARP in (b)–(f), respectively.

more efficient in compressing single slices but also enables information sharing across slices.

Figure 7 shows the reconstructed images via CARP, JPEG, and JPEG2000 of the same slices in the original image. The zoomed region in the last column in the coronal view shows CARP preserves substantially more details in the reconstruction compared to JPEG and JPEG2000.

4.3. YouTube video dataset

We use the YouTube dataset in [2], 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 repotting 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. Some frames of the sampled videos are displayed in Figure 8. Note that these YouTube videos have a low resolution of 256 by 256, and thus they favor the MPEG-4 standard as MPEG-4 is optimized at low bit-rate video communications [20].

CARP is applicable for streaming data by taking the entire video as input, treating time as an additional dimen-

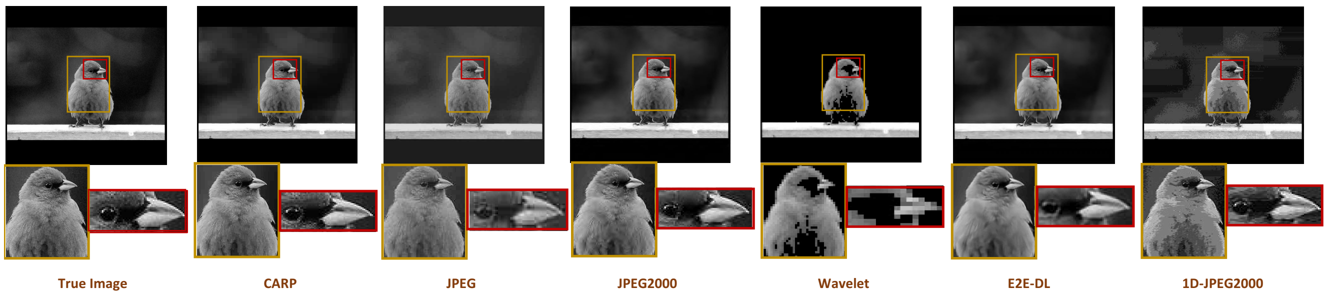


Figure 6. ImageNet: the comparison of reconstructed 2D bird image among six different methods when compression ratio is set to 30.

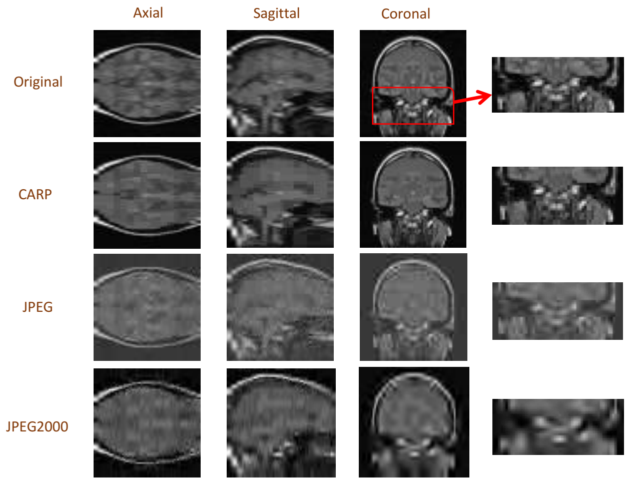


Figure 7. 3D brain image: compression performance comparison among CARP, JPEG, JPEG2000.

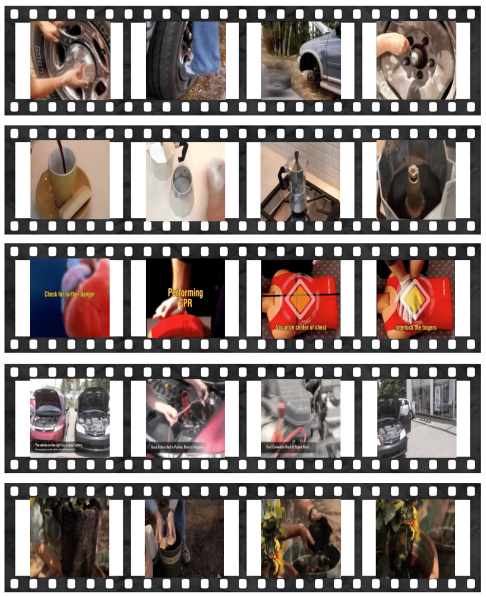


Figure 8. Selected frames of videos in the YouTube dataset.

sion, thus constituting a genuine video compression method like MPEG-4. In addition to the popular MPEG-4 standard for video compression, we also consider using JPEG and JPEG2000 through a frame-by-frame compression as in Section 4.2. Figure 9 presents the PSNR ratio curves (alternative methods over CARP) at various compression ratios for all the 100 videos, as well as the PSNR curve of CARP. CARP substantially outperforms methods of JPEG and JPEG2000 for nearly all individual videos at all compression ratios. CARP is better than MPEG-4 on all videos when the compression ratio is smaller than 10; for compression ratios between 10 and 20, MPEG-4 and CARP each performs better at a subset of the videos; for compression ratios above 20, CARP increasingly outperforms MPEG-4 at more videos. Moreover, CARP never underperforms MPEG-4 much on any individual image with the maximum PSNR ratio around 1.1. We note again that all the videos are at a low resolution that substantially favors MPEG-4. Overall, CARP gives better average PSNR than MPEG-4 at all compression ratios, as shown in Figure 1(c).

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 10. 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), while the regions reconstructed via the other three methods are more blurry.

4.4. Video: 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, shown in Figure 11. 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 12 plots the PSNR ratio curves (alternative method over CARP) among all the 180 videos as well as the PSNR curve for CARP at various compression ratios. We can see that CARP gives the best PSNRs uniformly for all videos at all compression ratios (up to 300).

For visual comparison, we randomly select one video and

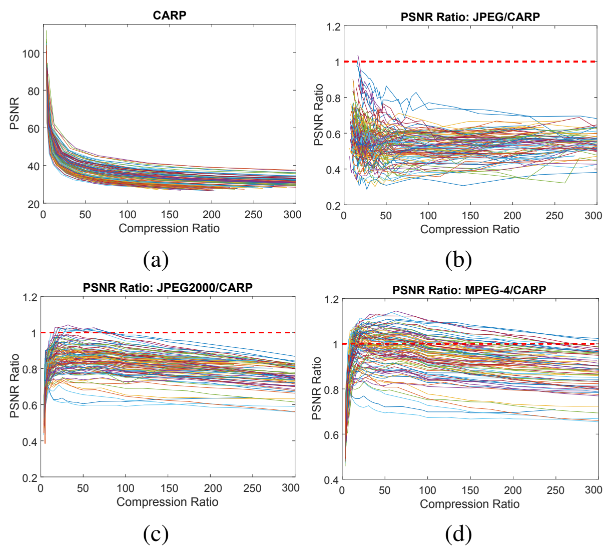


Figure 9. YouTube videos: PSNR curves of CARP for 100 videos in (a) and PSNR ratio curves of JPEG, JPEG200, MPEG-4 relative to CARP in (b)–(d), respectively.

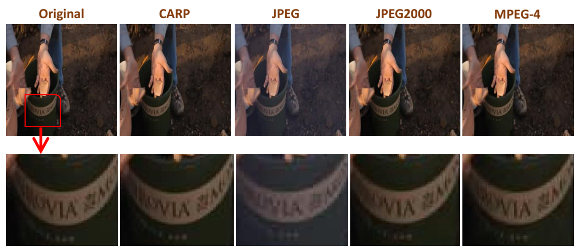


Figure 10. One frame of reconstructed YouTube videos. Left to right: original, CARP, JPEG, JPEG2000, and MPEG-4. Compression ratio is set to 30.

Figure 11. Selected frames from the surveillance video.

compare one frame of reconstructed videos. Figure 13 shows the original frame and reconstructed frames by CARP, JPEG, JPEG2000, 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 captured most details of the region in the original frame, while the regions reconstructed via the other three methods are more blurry (e.g., the edge of the yellow arrow).

5. Discussion

CARP uses a principled Bayesian hierarchical model to learn an optimal permutation on the image space, which effectively allows adaptive wavelet transforms on the image. It is computationally efficient in that it scales linearly with the number of pixels of an image. Currently for the 2D ImageNet data, the average computation time under our implementation of CARP, without any parallel computing, is around 3.17 second/image, while 0.82 second/image for JPEG, 0.40 second/image for JPEG2000, 0.33 second/image for Wavelet, 88.75 second/image for E2E-DL, and 0.44 second/image for 1D-JPEG2000, 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. We plan to implement a GPU parallelized version of CARP in the future to achieve substantial speedup.

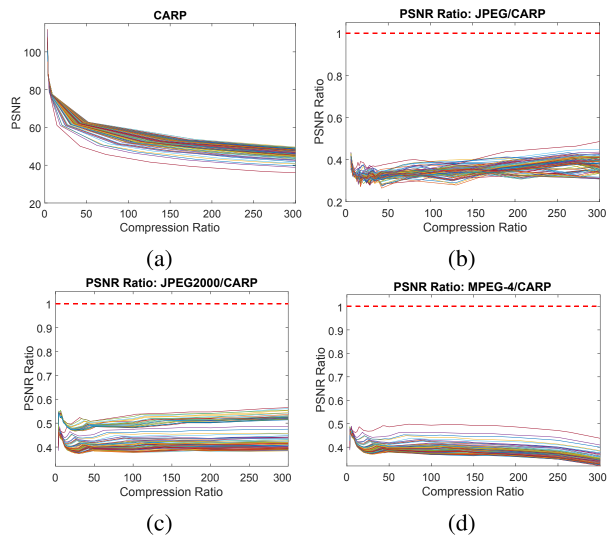


Figure 12. Surveillance video: PSNRs of CARP for 180 images in (a) and PSNR ratio curves for JPEG, JPEG200, MPEG-4 relative to CARP in (b)–(d), respectively.

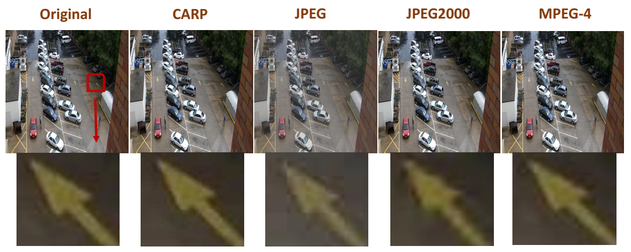


Figure 13. Surveillance video: the comparison of reconstructed surveillance video among four different methods when compression ratio is set to 30.

Acknowledgements

This work was partly supported by NSF Grant DMS-1749789.

References

[1] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017. 2

[2] Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Unsupervised learning from narrated instruction videos. In *Computer Vision and Pattern Recognition*, 2016. 5, 6

[3] Satish Anila and Nanjundappan Devarajan. The usage of peak transform for image compression. *International Journal of Computer Systems Science & Engineering*, 2(11):6308–6316, 2010. 3

[4] Mohammad Baig, Vladlen Koltun, and Lorenzo Torresani. Learning to inpaint for image compression. In *Advances in Neural Information Processing Systems*, pages 1246–1255, 2017. 2

[5] Johannes Ballé, Valero Laparra, and Eero Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, 2017. 2, 5

[6] Philip Brown, Tom Fearn, and Marina Vannucci. Bayesian wavelet regression on curves with application to a spectroscopic calibration problem. *Journal of the American Statistical Association*, 96(454):398–408, 2001. 4

[7] Jie Chen, Ut-Va Koc, and Ray Liu. *Design of Digital Video Coding Systems: A Complete Compressed Domain Approach*. CRC Press, 2001. 3

[8] Hugh Chipman, Eric Kolaczyk, and Robert McCulloch. Adaptive Bayesian wavelet shrinkage. *Journal of the American Statistical Association*, 92(440):1413–1421, 1997. 4

[9] Merlise Clyde and Edward George. Flexible empirical Bayes estimation for wavelets. *Journal of the Royal Statistical Society: Series B*, 62(4):681–698, 2000. 4

[10] Matthew Crouse, Robert Nowak, and Richard Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46(4):886–902, 1998. 4

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 5

[12] Iain Johnstone and Bernard Silverman. Empirical Bayes selection of wavelet thresholds. *Annals of Statistics*, 33(4):1700–1752, 2005. 4

[13] Meng Li and Li Ma. Warp: Wavelets with adaptive recursive partitioning for multi-dimensional data. *arXiv preprint arXiv:1711.00789*, 2017. 3, 4, 5

[14] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3223, 2018. 2

[15] Haojie Liu, Tong Chen, Qiu Shen, Tao Yue, and Zhan Ma. Deep image compression via end-to-end learning. In *Conference on Computer Vision and Pattern Recognition*, 2018. 2

[16] Li Ma. Adaptive testing of conditional association through recursive mixture modeling. *Journal of the American Statistical Association*, 108(504):1493–1505, 2013. 4

[17] Mathworks. Exploring slices from a 3-dimensional MRI data set. www.mathworks.com/help/images/exploring-slices-from-a-3-dimensional-mri-data-set.html, 2019 (accessed Nov 13, 2019). 5, 6

[18] David Minnen, Johannes Ballé, and George Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018. 2

[19] Jeffrey Morris and Raymond Carroll. Wavelet-based functional mixed models. *Journal of the Royal Statistical Society: Series B*, 68(2):179–199, apr 2006. 4

[20] Thomas Sikora. Mpeg-4 very low bit rate video. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 1440–1443, June 1997. 6

[21] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001. 2

[22] Roberto Vezzani and Rita Cucchiara. Video surveillance online repository (visor): an integrated framework. *Multimedia Tools and Applications*, 50(2):359–380, 2010. 5, 7

[23] Gregory Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), 1992. 2

[24] John Watkinson. *The MPEG Handbook*. Routledge, 2012. 3

[25] Wing Wong and Li Ma. Optional Pólya tree and Bayesian inference. *Annals of Statistics*, 38(3):1433–1459, 2010. 4