Online Graph Based Transforms for Intra-Predicted Imaging Data

D. Roy, T. Guha, V. Sanchez

PII: S0031-3203(25)01312-3
DOI: https://doi.org/10.1016/j.patcog.2025.112649
Reference: PR 112649

To appear in: *Pattern Recognition*

Received date: 27 February 2024
Revised date: 20 October 2025
Accepted date: 21 October 2025

Please cite this article as: D. Roy, T. Guha, V. Sanchez, Online Graph Based Transforms for Intra-Predicted Imaging Data, *Pattern Recognition* (2025), doi: https://doi.org/10.1016/j.patcog.2025.112649

Highlights

- A novel online graph-based transform learning framework (GBT-ONL) is proposed, which learns Graph-Based Transforms dynamically during encoding without any offline training phase.
- The method employs a shallow fully connected neural network (FC-NN) that predicts the graph Laplacian for each residual block as data is pro- cessed, enabling real-time adaptation to content variations.
- GBT-ONL requires no transmission of graph or model parameters in the bitstream since the same online learning process is mirrored at the decoder side.
- By learning transforms online, the framework eliminates the need for large, curated, or domain-specific training datasets, ensuring robustness and gen- eral applicability across different imaging domains (e.g., natural, medical, remote sensing data).

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2025) 1–23

Procedia Computer Science

Online Graph Based Transforms for Intra-Predicted Imaging Data

D. Roy[a,*], T. Guha[b], V. Sanchez[c]

[a] *School of Informatics and Digital Engineering, Aston University, United Kingdom*
[b] *School of Computing Science, University of Glasgow, United Kingdom*
[c] *Department of Computer Science, University of Warwick, United Kingdom*

Abstract

Orthogonal transforms are key components of several image and video compression systems and standards, as they provide a de-correlated representation of signals to enhance compressibility. However, the most commonly used transforms for compression, such as the Discrete Cosine transforms (DCT) and Discrete Sine transforms (DST), are fixed and non-adaptive, limiting their ability to capture complex or varying signal characteristics. Graph-based transforms (GBTs) have shown improved energy compaction and reconstruction performance, but face two major limitations: the need to signal graph information in the compressed bitstream, which increases overhead and may complicates decoder synchronization, and a dependency on offline training process, which is highly dependent on the quality and completeness of the training data. To address these issues, this paper introduces a novel framework, GBT-ONL, which learns GBTs online in the context of block-based predictive transform coding. The proposed GBT-ONL framework uses a shallow fully connected neural network to predict the graph Laplacian needed for both the forward and inverse GBT. By relying only on information available during encoding, GBT-ONL eliminates the need to signal additional information in the compressed bitstream, and removes the requirement for any prior offline training. Evaluations on several video sequences show that GBT-ONL outperforms both traditional (non-learnable) transforms and existing learnable transforms in terms of energy compaction, reconstruction error, and compression efficiency, as measured by BD-PSNR and BD-Rate metrics.

Keywords: Graph-based transform, GBT-ONL, video coding, compression, online training, predictive transform coding.

1. Introduction

Block-based Predictive Transform Coding (PTC) is a technique used by several image and video compression systems to improve performance, including the state-of-the-art compression techniques as defined by the High Efficiency Video Coding (HEVC) standard [1], Video Platform (VP9) [2], AOmedia Video 1 (AV1) [3] and the Versatile Video Coding (VVC) standard [4]. Fig. 1 shows a typical encoder-decoder pipeline for video compression based on block-based PTC and intra-prediction; i.e., the prediction is performed using information contained in the frame

*Corresponding author
Email addresses: D.Roy@aston.ac.uk (D. Roy), tanaya.guha@glasgow.ac.uk (T. Guha), v.f.sanchez-silva@warwick.ac.uk (V. Sanchez)

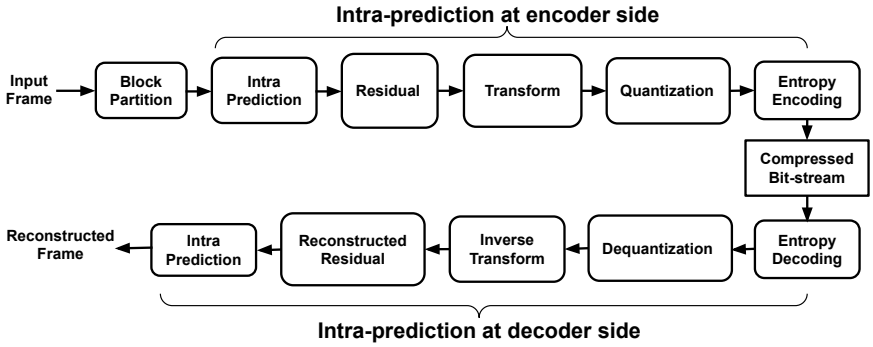


Figure 1. A typical encoder-decoder pipeline for video compression using block-based PTC using intra-prediction. The encoder side is in charge of compressing the frames, while the decoder side is in charge of decompressing them.

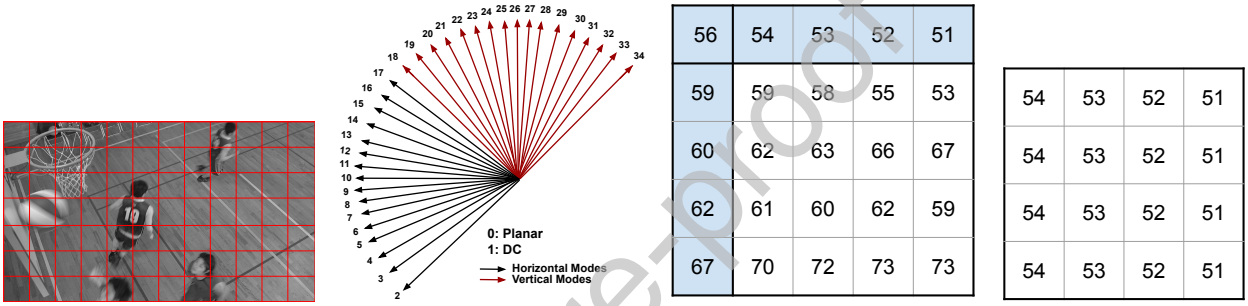


Figure 2. (a) Partition of a frame into non-overlapping blocks. (b) Directions of the intra-prediction modes that are common to the VVC and HEVC standards. (c) Sample block with the reference samples used for prediction shaded in blue. (d) Block predicted by the mode 26 (pure vertical mode) for the block in (c). Notice how the reference samples located above the block are simply copied into the locations of the predicted block following a vertical direction.

being predicted. In this figure, four main components can be distinguished: (i) prediction, (ii) transform, (iii) quantization, and (iv) entropy coding. In the prediction component, block-based PTC first divides the frame into several non-overlapping blocks (see Fig. 2 (a)) and then processes the blocks one by one following a specific order; e.g., a raster scanning order. It predicts each block by using previously processed blocks to exploit spatial redundancies. It computes a residual block for each block as the difference between the original and predicted block. In the transform and quantization components, each residual block is first transformed and the transform coefficients are quantized. In the final entropy encoding component, the quantized coefficients are encoded to produce a compressed bit-stream. The compressed blocks are reconstructed during encoding so that these blocks can be used to predict subsequent blocks. This allows replicating the process at the decoder when decompressing the bit stream.

PTC in state-of-the-art compression techniques uses one of several prediction modes to predict a block using intra-prediction. For example, the HEVC standard includes 33 angular modes to model 33 different directional patterns, and a DC mode and a Planar mode to predict smooth textures. The VVC standard, on the other hand, supports up to 87 prediction modes, which include 65 angular modes, 20 wide angular modes, a DC mode, and a Planar mode (see Fig. 2 (b)).

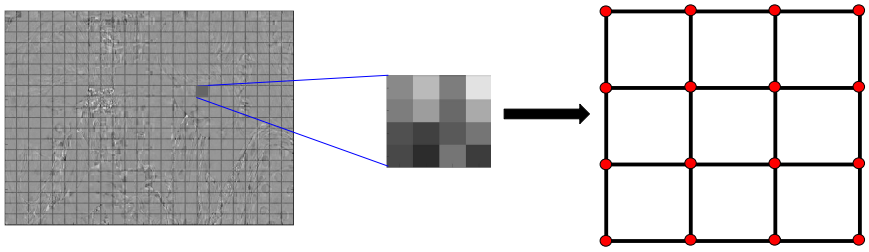


Figure 3. The residual signal of a frame partitioned using 4×4 blocks, with one of these blocks represented as a 4-connected graph with 16 nodes.

The transform in block-based PTC is essential for improving compression performance because it allows decorrelating the residual blocks into transform coefficients. An effective transform should have good energy compaction properties, i.e., it should capture most of the signal's energy in a few important transform coefficients. The Karhunen Loève Transform (KLT) is widely recognized as the linear transform with the best energy compaction properties for any arbitrary signal with a known covariance matrix. The KLT computes the eigendecomposition of the signal's covariance matrix. Since the Discrete Cosine Transform (DCT) does not require the signal's covariance matrix and its function bases are very similar to those of the KLT, it is widely used for the compression of natural images. However, the DCT is a fixed transform that does not account for the statistical properties of the residual block to be transformed. It has been shown that transforms that adjust to the statistical properties of the residual blocks can improve compression performance [5, 6].

Recently, the Graph-Based Transform (GBT) has been shown to attain promising results for data decorrelation and energy compaction [7], especially for block-based PTC. The GBT is an orthogonal transform that uses graphs to describe the signal to be transformed. Hence, the GBT can adapt to the signal's characteristics. In other words, a separate graph can be used on each residual block (see Fig. 3) to adequately represent the intrinsic structure and correlation among the residual values [8]. The GBT is constructed by eigendecomposition of the graph Laplacian of the residual block to be encoded. In [9, 10], it was shown that the GBT can outperform the DCT and the combination of the DCT and the Discrete Sine Transform (DST) as used in modern video codecs, in terms of energy compaction properties and reconstruction quality.

In general, when the GBT is used in block-based PTC, the same graph used to compute the GBT during the transform component should be available at the reconstruction stage to compute the inverse GBT needed to recover the residual block. This extra information should then be signaled into the compressed bitstream, hence increasing the overhead and hindering compression performance. To address this issue, several works have attempted to learn optimal GBTs by using machine learning (ML) [11]. Recently, in [12, 13] we propose a framework for learning GBTs by using deep learning (DL), where a trained fully connected neural network (FC-NN) is assumed to be common knowledge between the encoder and decoder of a compression system that uses block-based PTC with intra-prediction. As a result, that solution does not require signaling any additional information into the compressed bit-stream. However, it does require first training an FC-NN offline with the appropriate training data; i,e., training the FC-NN before

operation.

The idea of learning GBTs offline for compression purposes by using ML has gained increasing popularity recently [14]. However, the performance of such ML-based solutions depends on the amount, quality, and relevance of the training data [15]. In many cases, the training data may not accurately reflect the characteristics of the data to be processed after training the models. In such cases, the performance tends to be poor because, once trained offline, the models cannot adapt to the characteristics of new data. Hence, online optimization of the ML model [16] has emerged as an attractive solution to avoid the offline training process. In this context, online optimization refers to training the ML model as data is being processed.

Let us recall that in block-based PTC, blocks are processed sequentially following a specific order. Hence, online optimization is very amenable to be used within block-based PTC. In this work, we leverage online optimization and propose the GBT-ONL framework to learn GBTs online for block-based PTC within the context of intra-prediction. Specifically, our framework predicts the graph Laplacian needed to compute the GBT of each residual block by using a shallow FC-NN that is optimized online with the data being processed.

Since the GBT-ONL framework processes the data progressively to update the FC-NN's parameters as new data becomes available, it can learn GBTs that adapt to pattern changes in video frames. Hence, a key advantage of the online optimization used by the GBT-ONL framework is the dynamic adaptation to scene complexity, illumination and motion changes, as a model is optimized on each residual block of a video frame. Other important advantage is the fact that there is no need to signal any additional information into the compressed bitstream, as the same learning process used by the encoder can be replicated by the decoder using only the available reconstructed blocks. This contrasts with approaches based on pre-trained models, which require signaling the learned parameters into the bitstream, thus increasing overhead. Additionally, the online optimization used by the GBT-ONL framework removes the dependency on large, curated training datasets, making it more practical and adaptable across diverse content, including medical and remote sensing imaging data [17, 18, 19, 20, 21].

In summary, our work has two main contributions. First, it introduces an online optimization framework to compute GBTs, where an FC-NN is trained as blocks are being processed sequentially. This allows the FC-NN to adapt to each residual block to accurately predict the graph needed to compute the corresponding GBT. Second, since the training process is performed online using information available to block-based PTC, it can be exactly replicated at the decoder, thus avoiding the need to signal extra information to compute the inverse GBT for reconstruction. To the best of our knowledge, no other solutions have been proposed before to predict a graph Laplacian online for GBTs by using ML within the context of block-based PTC and without the need to increase the signaling overhead of the compressed bitstream.

Our performance evaluations on several video frames show that the GBTs learned by the GBT-ONL framework outperform the DCT and the DCT/DST in terms of the percentage of the signal's energy preserved by a small subset of the largest transform coefficients, the mean squared error of the reconstructed data, and the compression efficiency as measured by the BD-PSNR and BD-Rate metrics.

The rest of the paper is organized as follows. Section 2 summarizes the computation of the GBT and briefly reviews the related work on learning transforms offline within the context of block-based PTC. We describe our proposed GBT-ONL framework in Section 3. Section 4 presents our performance evaluations and discusses the results. Finally, Section 5 concludes this paper.

2. Related work

In this section, we first describe how GBTs are computed for residual blocks, followed by a summary of how GBTs are used within the context of block-based PTC. We then review several works that attempt to learn GBTs offline, including relevant works that attempt to learn offline other transforms used for image and video compression.

2.1. GBTs for residual blocks

The GBT of a (square) residual block $\mathbf{S} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$ with N residual values is usually constructed by eigendecomposition of the graph Laplacian, $\mathbf{L}$, of its undirected graph $G = (V, E, \mathbf{A})$. In this context, V is the set of N nodes $V = \{v_n\}_{n=1}^{N}$, E is the set of edges, and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the symmetric weighted adjacency matrix. Each node in V represent a pixel location in the residual block. The entry $\mathbf{A}_{ij}$ in $\mathbf{A}$ represents the weight of the edge e_{ij} connecting nodes v_i and v_j, where $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ and nodes v_i and v_j represent, respectively, pixel locations i and j, in the block. If there is no edge $e_{i,j}$ connecting nodes v_i and v_j, $\mathbf{A}_{ij} = 0$. Large values in $\mathbf{A}$ usually represent a high similarity between the connected nodes, according to a given criterion, for example, similarity between residual values. The graph Laplacian, $\mathbf{L}$, is computed as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix, whose n^{th} diagonal element is equal to the sum of the weights of all edges incident onto node v_n. The eigendecomposition of $\mathbf{L}$ is used as the orthogonal transform for the residual block since it has a complete set of eigenvectors with real, non-negative eigenvalues.

Let us denote the eigendecompostion of $\mathbf{L}$ by $\{\lambda_q, \mathbf{u}_q\}$ where $\lambda_q/\mathbf{u}_q$ is the q^{th} eigenvalue/eigenvector pair and $\mathbf{U}$ is the set of eigenvectors. Analogous to the classical Fourier transform, one can define the GBT for the signal $\mathbf{S} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$ represented by the graph G, as the expansion of $\mathbf{S}$ in terms of the eigenvectors of $\mathbf{L}$:

$$\hat{\mathbf{S}}(\lambda_q) = \langle \mathbf{S}, \mathbf{u}_q \rangle = \sum_{k=0}^{N-1} \mathbf{S}(k)\mathbf{u}_q(k) = \mathbf{F}\mathbf{S}, \tag{1}$$

where N is the number of nodes, $\mathbf{F} = \mathbf{U}^{-1}$ is the graph Fourier transform and $\sigma(\mathbf{L}) = (\lambda_0, \lambda_1, \cdots, \lambda_{N-1})$ denotes the set of eigenvalues of $\mathbf{L}$. The signal can be reconstructed by the inverse GBT, which is given by $\mathbf{S} = \mathbf{F}^{-1}\hat{\mathbf{S}} = \mathbf{U}\hat{\mathbf{S}}$. As a graph is defined by an adjacency matrix, $\mathbf{A}$, it is possible to generate different transforms for the same block by using different graph topologies and weights in G [22]. In general, the graph connectivity and the edge weights are inferred from the data to be transformed.

2.2. GBTs within the context of PTC

Within the context of block-based PTC, the GBT performs significantly well in generating decorrelated coefficients that can compact the signal's energy into a few largest coefficients [23]. S. Bagheri *et.* al [24] propose a solution for

learning a graph Laplacian by minimizing a graphical lasso. In their work, the transformation of a signal is realized by a hybrid model that combines the KLT and the DST. In [25], the authors show that GBTs are optimal for block-based PTC if the signal follows a Gaussian Markov Random Field (GMRF) model. In [26], the authors propose an optimized GBT for residual signals by introducing an intra-prediction scheme that exploits the differences between neighboring pixel pairs and the mean of a cluster of pixels. In our previous work [9], we propose a class of GBTs for PTC that are constructed by using a graph with unit edge weights and weighted self-loops in every node. The self-loops have been shown to improve the decorrelation properties of the GBT. To avoid signaling any additional information into the bitstream to compute the inverse GBT, we also introduce a coding framework that uses a template-based strategy to predict blocks in the pixel and residual domains. Similarly, the work in [10] eliminates the need to signal the graph information into the bitstreamby computing the GBT based on a predicted residual. Recently, the work in [27] introduces the Symmetry-based Graph Fourier Transforms (SBGFTs), which are non-separable transforms based on the generation of symmetric graphs by adding symmetrical connections between nodes. Although the SBGFTs do not explicitly adapt to data, the authors exploit the correlations between optimal graphs and prediction modes in intra-prediction to provide a type od data adaptation.

2.3. Offline and online learning of GBTs

Several works that attempt to learn GBTs within the context of block-based PTC have been proposed. In [28], the author proposes two different techniques to design GBTs. In the first technique, they formulate an optimization problem to learn graphs from data and provide solutions for optimal separable and non-separable GBT designs, called GL-GBTs. The optimality of the proposed GL-GBTs is also theoretically analyzed based on GMRF models for residual signals. The second technique develops edge-adaptive GBTs (EA-GBTs) to adapt transforms to signals with strong edges (discontinuities). To accomplish this task, they train a large model offline with a large dataset collected by predicting blocks of several sizes with several intra-prediction modes. The advantages of EA-GBTs are both theoretically and empirically demonstrated. The experimental results show that their proposed transforms can outperform the KLT. In [11], the authors propose the graph template transform (GTT), which approximates the KLT by exploiting a priori information the about signal as represented by a graph template. The GTT, which is learned by a constrained optimization framework, has been shown to achieve a rate-distortion performance similar to that of the KLT with significantly less complexity. In [29], the authors introduce the GBSTs (Graph Based Separable Transforms), based on two line graphs with optimized weights. For the construction of the GBST, the authors formulate a graph learning problem to design two separate line graphs using row-wise and column-wise residual block statistics, respectively. They analyze the optimality of the resulting separable transforms and show that the GBST can outperform the DCT, the DST, and the separable KLT. In [30] authors proposed to combine deep Probabilistic Graphical Networks (PGNs) and deep compression techniques together to derive sparse versions of the deep probabilistic models. In [31], we propose a GBT based on 3D convolutional neural networks (GBT-CNN). The 3D convolutional neural network (3D-CNN) predicts the graph information needed to compute the transform and its inverse, thus reducing the signaling cost

to reconstruct the data after transformation. We show that the GBT-CNN can outperform the DCT and the DCT/DST in terms of the percentage of energy preserved by a small subset of the largest transform coefficients, the mean squared error of the reconstructed data, and the transform coding gain. In [32], the authors address a problem of learning graph Laplacians by adopting a factor analysis model that enforces minimizing the variations of the signal on the learned graph. The work in [33] reports improvements over the DCT by introducing a novel adaptive separable path that can provide better data decorrelation for intra-predicted data. However, differently from our proposed GBT-ONL framework, it increases the signaling overhead in the bitstream, as it requires indicating the usage of the GBT on each block.

Recently, the work in [34] introduces a block-adaptive separable path graph-based transform (GBT), which focuses on adaptively modifying the block size and learning a GBT. Although the GBT is alo learned in an online scenario, the learning process relies on sequential K-means clustering, where each available block size has K clusters and K GBT kernels. Hence, different from our GBT-ONL framework, which optimizes an FC-NN, their approach relies on finding the nearest cluster to a block, which is then used to derived the GBT.

2.4. *Offline learning of other transforms*

Apart from the GBT, several methods to learn other transforms for compression purposes may be found in the literature [35]. For example, in [36], the authors propose a fully unsupervised deep-learning framework that can extract a meaningful and sparse representation of raw high-frequency signals by embedding important properties of the fast discrete wavelet transform (FDWT) in the architecture. In [37], a novel sub-pixel convolutional generative adversarial network (GAN) is learnt for reconstruction of images in compressed sensing. Since the DCT is widely used for block-based PTC, many works attempt to learn the mapping relationship between images compressed by using the DCT and their original version to reduce compression artifacts [38, 39]. Similarly, several works focus on learning the KLT offline [40]. For example, in [41] the authors propose a novel signal-independent separable transform based on the KLT to improve its efficiency in block-based PTC. A KLT matrix is trained offline using several residual blocks to account for different residual characteristics. Deep learning models are also exploited for learning the KLT [42] under a supervised training framerwork. Recently, the use of rate distortion optimized transform (RDOT) designs has re-gained popularity to create data-dependent transforms that minimize rate-distortion costs. This is commonly achieved by using a clustering step to identify training data examples that cannot be well represented by a fixed transforms; e.g., the DCT. For example, the work in [43] proposes using this rate distortion approach to learn separable GBTs.

To summarize, a key difference between the work presented in this paper and the previously proposed methods that attempt to learn an orthogonal transform is that several of those methods need to train a model offline and thus depend on the quantity, quality, and relevance of the training data. Those methods that rely on an online learning process, employ clustering approaches to find the most similar cluster to a block, which is subsequently used to construct a GBT. Our proposed GBT-ONL framework does not require an offline training process, and more importantly, relies on a true online learning process that requires the optimization of a loss function, as explained next.

3. Proposed GBT-ONL framework for intra-predicted data

Online optimization aims to learn a mapping function based on a sequence of samples as the samples are processed by a model [44, 45, 46]. Such a mapping function is expected to perform a specific task based on the processed samples, for example, classification or regression. In the case of online optimization of an FC-NN, the mapping function is learned by defining the parameters of the network as samples are being processed. Those parameters are expected to perform the task very well for the current sample. Specifically, the parameters are usually initialized to random values and are updated sequentially using only the data being processed. To improve performance, the parameters learned for the current sample can be used as the initial set of parameters to be optimized for the next sample.

Our work concentrates on learning GBTs online within the context of block-based PTC. We focus on intra-prediction as currently performed by the HEVC and VVC standards; however, this work is codec-agnostic and can be used with any video and image codec that uses block-based PTC with intra-prediction. More specifically, our GBT-ONL framework relies on a shallow FC-NN trained over several iterations of gradient descent (GD) to predict the graph Laplacian required to compute the GBT for the current residual block. Once the optimization for the current residual block is complete, the FC-NN is optimized to predict the graph Laplacian of the subsequent residual block using as the initial set of parameters those optimized for the previous block. This process is repeated until all residual blocks are processed. The process of defining the initial set of parameters to be optimized for the current block can be expressed mathematically for any two consecutive blocks with indices k and $k+1$ as follows:

$$\mathbf{W}_{k+1}^{0} \leftarrow \tilde{\mathbf{W}}_{k}, \tag{2}$$

where $\mathbf{W}_{k+1}^{0}$ and $\tilde{\mathbf{W}}_{k}$ denote the initial and final set of parameters for block $k+1$ and block k, respectively. For the first residual block of a frame, our shallow FC-NN uses parameters initialized to known values, i.e., $\mathbf{W}_{k=0}^{0}$ is known for block $k=0$. Moreover, the FC-NN uses information obtained from the blocks that have been already processed by block-based PTC as the input. Consequently, no additional information needs to be signaled into the bitstream to repeat the same optimization process during the reconstruction of the blocks since the same input is available when blocks are reconstructed sequentially and in the same order used to compute their GBTs. Let us recall that such sequential encoding and decoding processes are common in modern video and image codecs that use block-based PTC.

By using online optimization, our main objective is to learn a mapping function, $\mathtt{f}(\cdot)$, between an *average residual block* for the current block k, denoted by $\mathbf{C}_k$, and the graph Laplacian of such an average residual block:

$$\hat{\mathbf{L}}_{k} \approx \mathtt{f}(\mathbf{C}_{k}), \tag{3}$$

where $\hat{\mathbf{L}}_k$ is the predicted graph Laplacian of the average residual block and $\mathbf{C}_k = \frac{1}{3}\sum_{d=1}^{3}\mathbf{M}_d$ is computed as the

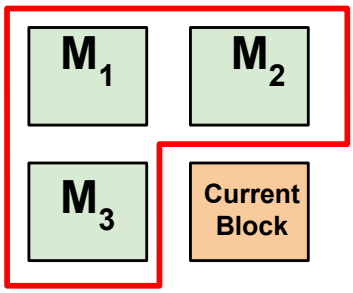


Figure 4. Surrounding residual blocks of the current residual block to be encoded.

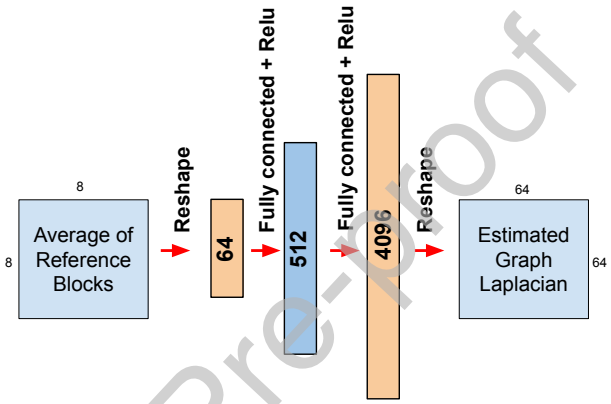


Figure 5. Architecture of the shallow FC-NN used by the proposed GBT-ONL framework for 8×8 blocks.

average residual of the three residual blocks surrounding the current block, k. Here, $\mathbf{M}_d$ represents the d^{th} surrounding residual block (see Fig. 4).

Note that in PTC, the blocks surrounding the current block, k, are already processed. Consequently, the same three residual blocks, $\{\mathbf{M}_d\}$, are available when reconstructing a frame block-by-block. Hence, they can be used as an input to the same FC-NN for predicting the same graph Laplacian, which is needed to compute the inverse GBT for block k. The rationale behind using these three surrounding residual blocks is based on their similarities with the current residual block. Namely, these three blocks are expected to have similar characteristics to those of the residual block to be transformed. For residual blocks located in the corner or along the edges of a frame or image, which may not be surrounded by three residual blocks, we use a residual block with a constant value equal to the DC value of the frame; e.g., for 8 bpp images, we use a value of 128.

Our solution to learn the mapping function in Eq. 3 is based on an encoding-decoding shallow FC-NN, as depicted in Fig. 5 for the case of 8×8 blocks. This shallow FC-NN has an input layer of 64 neurons and a 512-neuron hidden layer, leading to the output layer of 4096 neurons. The average residual block, $\mathbf{C}_k$, in vector form, denoted by $\mathbf{c}_k$ is used as input to the FC-NN, which is trained to predict the graph Laplacian of $\mathbf{C}_k$, also in vector form and denoted by $\mathbf{l}_{\mathbf{c}_k}$. In other words, $\mathbf{l}_{\mathbf{c}_k}$ serves as the *ground truth* for the training process. Under the assumption that the three

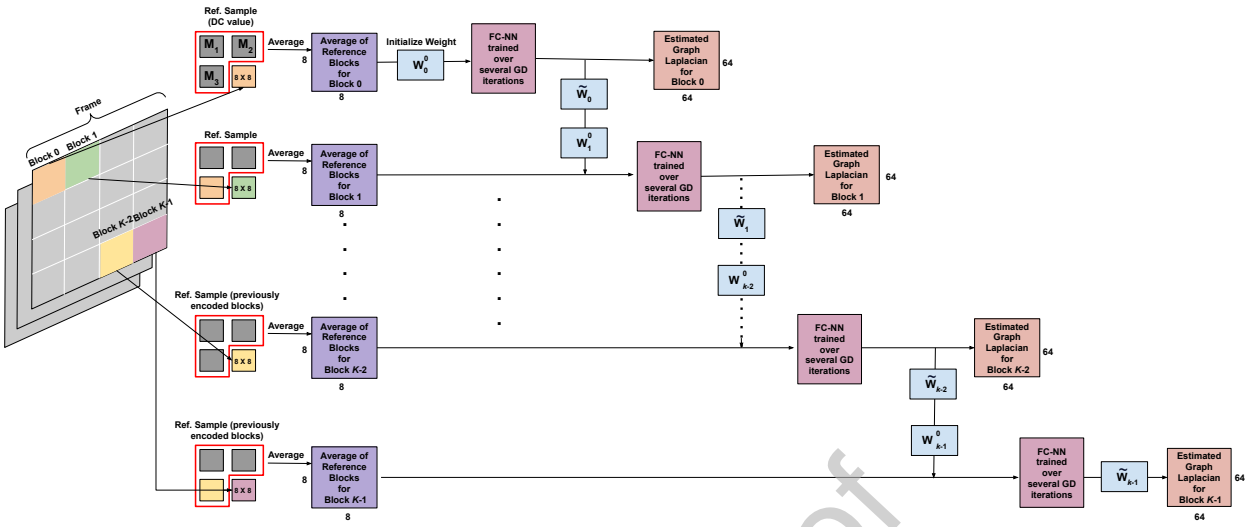


Figure 6. Sequential processing of blocks by the proposed GBT-ONL framework. To estimate the graph Laplacian of the current block, k, the shallow FC-NN uses the average residual block in vector form, denoted by $\mathbf{c}_k$.

surrounding residual blocks $\{\mathbf{M}_d\}$ are similar to the residual block to be transformed, the graph Laplacian $\mathbf{l}_{\mathbf{c}_k}$ is then expected to be similar to that of the current residual block, denoted by $\mathbf{l}_k$; hence $\mathbf{l}_{\mathbf{c}_k} \approx \mathbf{l}_k$.

Fig. 6 shows the overall functionality of the GBT-ONL framework to predict the graph Laplacian for each residual block k under block-based PTC using intra-prediction. Note that the shallow FC-NN is optimized for each block k over several iterations of gradient descent (GD) to accurately predict $\mathbf{l}_{\mathbf{c}_k}$. This optimization process is stopped based on threshold ξ, i.e., when the Mean Squared Error (MSE) between the predicted graph Laplacian, $\hat{\mathbf{l}}_{\mathbf{c}_k}$, and the corresponding ground truth, $\mathbf{l}_{\mathbf{c}_k}$, is less than ξ, or when enough iterations of GD have been performed. In other words, the FC-NN is overfitted on the input $\mathbf{c}_k$. Note that this overfitting process is appropriate for block-based PTC as the prediction is based on a single input. The weights found after optimizing the FC-NN on block k, i.e., $\tilde{\mathbf{W}}_k$, are used as the initial weights for block $k+1$ (see Eq. 2). The process is repeated for all K residual blocks in the frame.

Algorithm 1 [1] summarizes the online optimization process used by the GBT-ONL framework, where P is the maximum number of GD iterations to be performed for the current block, k, α is the learning rate, $Arc(\cdot)$ denotes the architecture of the shallow FC-NN, $\{DC_{value}\}$ is a reference block in vector form with all values equal to the DC value of the image, $\mathbf{m}_d$ is the d^{th} reference residual block in vector form, and $\{r, c\}$ denotes the row and column, respectively, where a block is located in the image The functionality of this algorithm is explained next:

- Line 1 of the algorithm iterates over all K residual blocks.
- Line 3 initializes all the parameters of the FC-NN for the first block to a value of 0.5.
- Line 4 calculates the average residual block for block $k = 0$ as a block with DC values.

[1] The implementation details are available on GitHub: https://github.com/debaleena82/GBT-ONL.git

Algorithm 1: Online training used by the GBT-ONL framework for an image with K blocks.

Require: $\{\{\mathbf{m}_d\}, \mathbf{l}_{\mathbf{c}_k}\}$ for each block k, $Arch(\cdot)$, P, α, ξ, $\{DC_{value}\}$

1: **for** $k = 0 \rightarrow (K-1)$ **do**
2: **if** $k = 0$ **then**
3: $\mathbf{W}^0_{k=0} = \{0.5\}$
4: $\mathbf{c}_k \leftarrow \{DC_{value}\}$
5: **else**
6: $\mathbf{W}^0_k \leftarrow \tilde{\mathbf{W}}_{k-1}$
7: **if** $r = 0$ AND $c! = 0$ **then**
8: $\mathbf{c}_k \leftarrow \frac{1}{3}(\mathbf{m}_3 + \{DC_{value}\} + \{DC_{value}\})$
9: **elseif** $r! = 0$ AND $c = 0$ **then**
10: $\mathbf{c}_k \leftarrow \frac{1}{3}(\mathbf{m}_2 + \{DC_{value}\} + \{DC_{value}\})$
11: **else**
12: $\mathbf{c}_k \leftarrow \frac{1}{3}\sum_{d=1}^{3} \mathbf{m}_d$
13: **end**
14: **end**
15: **for** $p = 1 \rightarrow P$ **do**
16: $\{\hat{\mathbf{l}}_{\mathbf{c}_k}, \mathbf{W}^p_k\} \leftarrow Arch(\mathbf{c}_k, \mathbf{l}_{\mathbf{c}_k}, \alpha, \mathbf{W}^0_k)$
17: **if** $||\hat{\mathbf{l}}_{\mathbf{c}_k} - \mathbf{l}_{\mathbf{c}_k}||^2_2 > \xi$ **then**
18: $\mathbf{W}^0_k \leftarrow \mathbf{W}^p_k$
19: go to line 16
20: **else**
21: $\tilde{\mathbf{W}}_k \leftarrow \mathbf{W}^p_k$
22: **return** $\hat{\mathbf{l}}_{\mathbf{c}_k}$
23: **end**
24: **end**
25: **end**

- Line 6 initializes the parameters of the FC-NN to the parameters found for the previous block.
- Lines 7 -13 initialize the average residual block according to the position of the current block k to account for any unavailable residual block $\mathbf{m}_d$.
- Line 15 optimizes the FC-NN for block k over a maximum of P iterations of GD.
- Line 16 computes the predicted graph Laplacian and the optimized parameters of the FC-NN for iteration p of GD, denoted by $\mathbf{W}^p_k$. The optimization is based on the following loss function:

$$\mathcal{L}(\hat{\mathbf{l}}_{\mathbf{c}_k}, \mathbf{l}_{\mathbf{c}_k}) = || \hat{\mathbf{l}}_{\mathbf{c}_k} - \mathbf{l}_{\mathbf{c}_k} ||^2_2 + \lambda || \mathbf{W}^p_k ||^2_2, \tag{4}$$

  where $|| \cdot ||_2$ is the L2 norm and λ is a hyperparameter to control the level of L2-regularization on $\mathbf{W}^p_k$.
- Line 17 computes the square of the error between the ground truth and the predicted graph Laplacian and checks

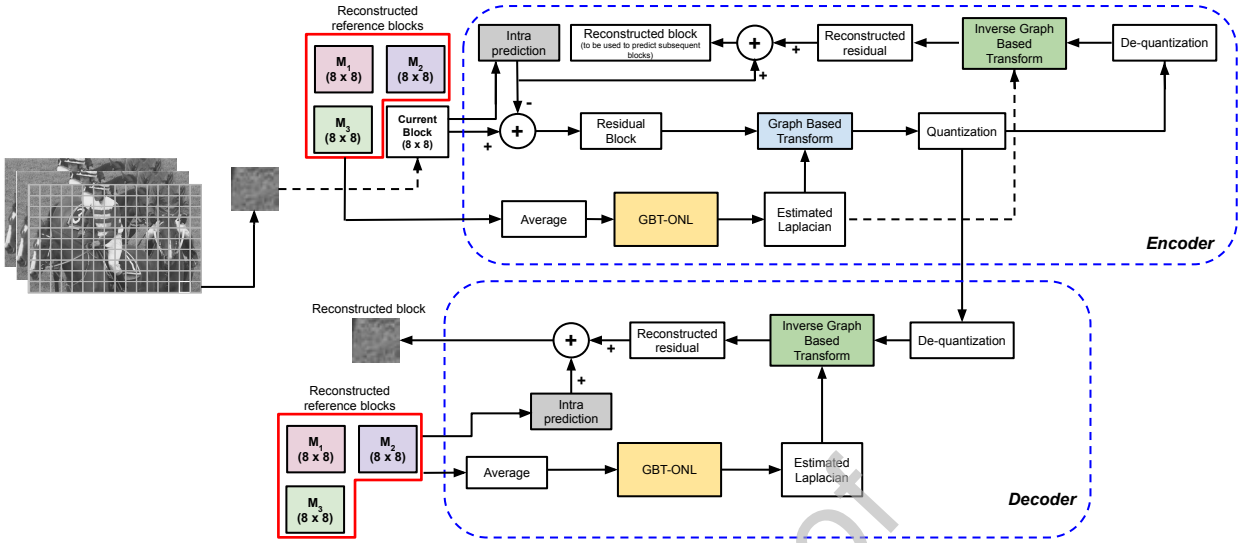


Figure 7. The GBT-ONL framework incorporated into an encoder-decoder system that uses block-based PTC for compression.

if this squared error is above the threshold ξ. If this squared error is above ξ, the parameters found after iteration p of GD are used as the initial set of parameters to be further optimized in Line 16. Otherwise, Line 21 defines the final set of parameters as those found at iteration p.

- Line 22 returns the predicted graph Laplacian, which is to be used to compute the GBT for block k.

Fig. 7 shows how the proposed GBT-ONL framework can be incorporated into an encoder-decoder pipeline that uses block-based PTC for compression with intra-prediction. Our framework assumes that the initial parameters $\mathbf{W}_{k=0}^{0}$ of the shallow FC-NN for block $k = 0$ are *common knowledge* between the encoder and decoder. Note that the residual blocks $\{\mathbf{M}_d\}$ used to compute the average residual block $\mathbf{C}_k$ are those available at the encoder after the corresponding blocks are processed and reconstructed. This guarantees that even after quantization of the corresponding transform coefficients, these residual blocks are the same as those available at the decoder.

4. Performance Evaluation

4.1. Datasets and experimental setup

For our experiments, we use the standard video sequences provided by the ISO/IEC JCT1/SC29/WG11 under the Common Test Conditions and Software Reference Configurations [47]. Our dataset to evaluate the performance of the GBT-ONL framework comprises several 4:2:0 YUV video sequences commonly used to test the performance of video codecs. These sequences are organized into six classes: A, B, C, D, E, and Screen Content (SC). They cover a wide range of characteristics in terms of length, scene complexity, and type of visual content. These sequences serve as a benchmark for evaluating transform performance (see Table 1). We use intra-prediction on each frame of

Table 1. Characteristics of the 4:2:0 video sequences used for evaluation.

Name	Resolution	No. of frames	Frame rate (fps)	Bit depth
Class A				
Traffic	2560×1600	150	30	8
People_on_street	2560×1600	150	30	8
Nebuta_festival	2560×1600	150	30	10
Class B				
Kimono	1920×1080	240	24	8
Cactus	1920×1080	500	50	8
Park_scene	1920×1080	240	24	8
BQTerrace	1920×1080	600	60	8
Class C				
Race_horse	832×480	300	30	8
BQMall	832×480	600	60	8
Party_scene	832×480	500	50	8
Basketball_drill	832×480	500	50	8
Class D				
Race_horse_D	416×240	300	30	8
Blowing_bubble	416×240	500	50	8
BQ_square	416×240	600	60	8
Basketball_pass	416×240	500	50	8
Class E				
Kristine_and_Sara	1280×720	600	60	8
Four_people	1280×720	600	60	8
Jhonny	1280×720	600	60	8
Class SC				
China_speed	1024×768	500	30	8
Slide_show	1280×720	500	20	8
Sc_Map	1280×720	600	60	8
Sc_Programming	1280×720	600	60	8

the video sequences. Specifically, we use blocks of 8×8 pixels and the 35 intra-prediction modes that are common to the HEVC and VVC standards. We use the mode that provides the lowest residual signal for block k. Note that modern video codecs usually make intra-prediction decisions based on the Y and G components of Y:U:V and RGB frames, respectively. For this reason, we use only these components of our dataset. For the evaluation of offline trained models that predict transforms, we use 61,440 samples (residual blocks) which are partitioned into 80% for training and 20% for testing, ensuring there is no overlap between the two sets. The residual data is processed using quantization parameters (QPs) of 22, 27, 32, and 37, which are standard values used to attain various compression levels. We evaluate performance using standard metrics, including BD-PSNR, BD-BR (with respect to DCT), and Mean Squared Error (MSE). These metrics are computed to assess both the reconstruction quality and compression efficiency. No additional pre-processing is applied beyond the quantization of residual blocks according to the selected QP values.

We compare the GBTs learned by the proposed GBT-ONL framework against other GBTs, which vary in terms of graph topology (including edge weigths) and construction. Specifically, we compare a GBT that uses covariance

matrices from several training examples to estimate the graph Laplacian, hereinafter called the GL-GBT [48]. Note that this GBT uses offline training. We also compare the GBT-CNN in [31], which uses graphs with unit edge weights and no self-loops in the vertices. This method relies on training a 3D-CNN offline. Our experiments also include other commonly used transforms for compression purposes; i.e., the KLT, the DCT, and the DCT/DST as used in the HEVC and VVC standards. The KLT is used as the baseline in our experiments, as it is expected to provide the best performance. However, it is important to recall that the KLT requires storing the covariance matrix of each block. All these transforms; i.e, KLT (baseline) , GL-GBT [28], GBT-CNN [49], DCT (fixed transform), DCT/DST (fixed transform) and our proposed GBT-ONL, are evaluated on the same data.

We organize the evaluated transforms into two groups: i) those that are computed based on offline training, ii) those that are computed based on online training (including ours), and iii) those that require no training at all (e.g., DCT). Since the GBT-ONL framework does not require offline training, we highlight in bold font and underlined font the best and second best results, respectively, among those obtained by the GBTs learned by the GBT-ONL framework, the DCT, and the DCT/DST.

For all experiments, we initialize the parameters of the FC-NN used by the GBT-ONL framework to a value of 0.5 for the first block of each frame; i.e., $\mathbf{W}^0_{k=0} = \{0.5\}$. To optimize the FC-NN, we use up to $P = 100$ iterations of GD with a threshold of $\xi = 1e^{-8}$. In other words, the optimization process for blokc k is stopped after $P = 100$ iterations of GD steps or if the MSE between the predicted and ground-truth graph Laplacian is less than or equal to $\xi = 1e^{-8}$. A learning rate $\alpha = 0.005$ is chosen based on several tests to ensure reaching threshold ξ without significantly increasing the computational complexity. A small learning rate results in longer times to process each block but may allow reaching more precisely the threshold ξ. A large learning rate, on the other hand, results in short processing times, but may not allow reaching precisely the threshold ξ. As explained in Section 3, the blocks are processed sequentially and the parameters optimized for block k are used as the initial set of parameters for block $k + 1$.

4.2. Reconstruction quality and energy compaction performance with unquantized coefficients

The efficiency of a transform is usually measured by the signal's energy that a small subset of the largest transform coefficients can concentrate. Based on this fact, we first compute the MSE of the reconstructed frames by using only a subset of the largest coefficients under the assumption that no quantization is applied to the coefficients. We also compute the percentage of the total signal's energy preserved (PE) by the set of coefficients used for reconstruction, where the PE values are computed as the average of the square of the coefficients used for reconstruction divided by the square of all coefficients. To select the coefficients used for reconstruction, we set a threshold that indicates the minimum absolute value that the coefficients should have to be used to reconstruct a block. We gradually decrease this threshold to increase the number of coefficients to be used for reconstruction. This strategy gradually includes the largest coefficients in a subset by gradually lowering an initial large threshold [50]. Fig. 8 shows a toy example of this strategy.

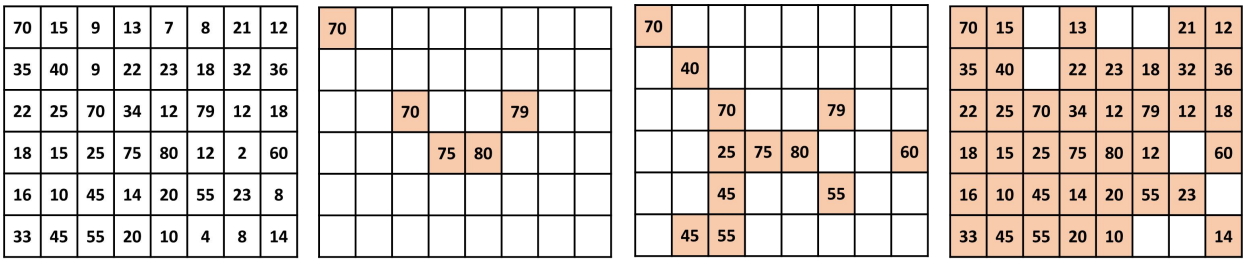


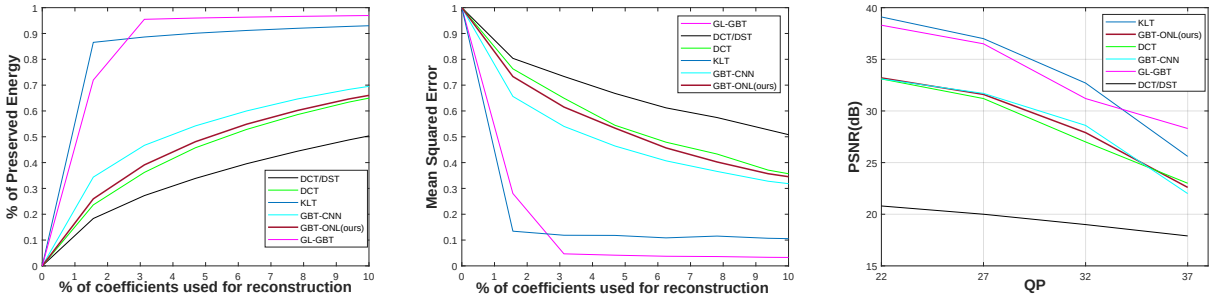
Figure 8. (a) Absolute values of transform coefficients of a sample block of 8×6. (b) Sub-set of coefficients selected for a threshold = 70. (c) Sub-set of coefficients selected for a threshold = 40. (d) The majority of the coefficients are in the sub-set when the threshold value is 10.

Table 2 tabulates the PE (%) and MSE values for all evaluated sequences using 5% and 10% of the largest coefficients, while Table 3 summarizes performance by tabulating average PE and MSE values over all evaluated sequences. Compared to the DCT, the transforms learned by the GBT-ONL framework preserve 3.13% more energy for Class A videos and 3.65% more energy for all classes, on average, if only 5% of the largest coefficients are used. The GL-GBT outperforms the KLT in terms of the PE preserved and the MSE for the natural images of Class D (see Fig. 9 (a)-(b) for an example). Recall that the GL-GBT and GBT-CNN rely on offline training, hence their performance depends on the amount and relevance of the training data whereas the proposed GBT-ONL framework adapts to the patterns of the data being processed without any offline training. As expected, the KLT performs the best for Classes C, E, SC. However, let us recall that the KLT requires knowledge of the covariance matrix of each residual block. This implies increasing the amount of data to be stored in the compressed bit-stream.

4.3. Reconstruction quality with quantized coefficients

We compute the reconstruction quality attained by the evaluated transforms in terms of the PSNR when quantization is used on the coefficients. Specifically, we employ four quantization parameters (QPs) used by the HEVC and VVC standards: $QP = \{22, 27, 32, 37\}$. Table 4 tabulates PSNR values for the evaluated videos when these QPs are applied to the transform coefficients, while Table 5 summarizes performance by tabulating average PSNR (dB) values over all evaluated sequences. Note that for Class A sequences and all QPs, except for $QP = 27$, the PSNR values attained by the transforms learned by the GBT-ONL framework are larger than those attained by the DCT. For example, the GBT-ONL framework outperforms the DCT and the DCT/DST by 2.1 dB and 7.2 dB, respectively, for the *Traffic* sequence (Class A) when $QP = 37$. On the other hand, for Class C and Class A, the GBT-ONL framework outperforms the DCT for $QP = 27$. Since these two classes include sequences depicting several smooth textures, the DCT performs very well on them. For Class D and $QP = \{22, 27, 32\}$, the GBT-ONL framework outperforms the DCT, while the DCT performs the best for $QP = 37$. For the sequence *BQsquare*, the DCT outperforms the GBT-ONL framework by only 0.4 dB (see Fig. 9 (c))

The most challenging videos are those in the SC class, which tend to contain several edges. Fig. 10 depicts a frame of the video *SC_Programming*. For this video, the GBTs learned by the GBT-ONL framework outperform the



(a) (b) (c)

Figure 9. (a) PE (%) and (b) MSE when up to 10% of the largest coefficients are used for reconstruction of the Class D video *BQsquare*. (c) PSNR (dB) of the Class D video *BQsquare* for several QPs.

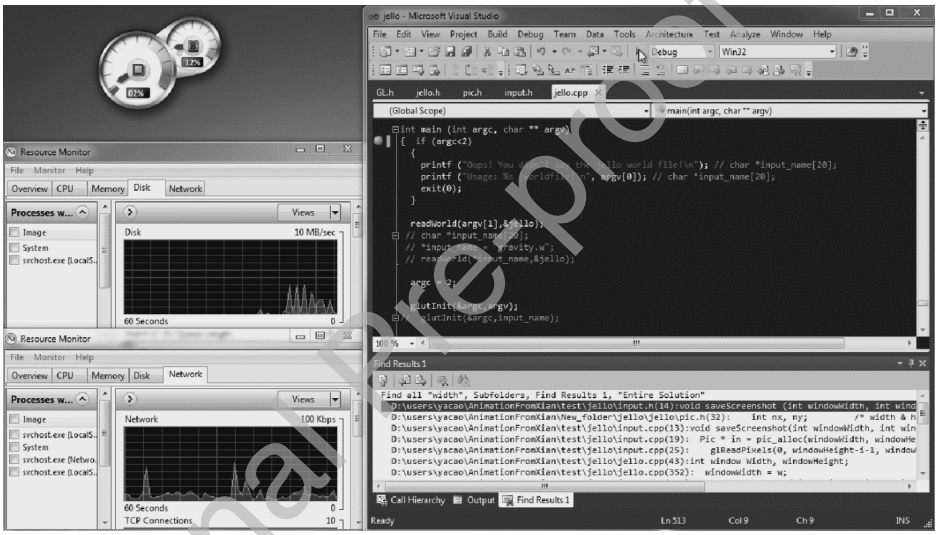


Figure 10. A frame of sequence *SC_Programming* of Class SC.

DCT by 2.4 dB, 2.1 dB, and 0.6 dB for $QP = \{22, 27, 32\}$, respectively. For all SC videos, the DCT surpasses the GBT-ONL framework for $QP = 37$.

On average, the GL-GBT outperforms the GBT-ONL framework by 1.8 dB for $QP = 22$ for all the evaluated video sequences since this method uses offline training to predict the graph Laplacian.

4.4. Compression performance with quantized coefficients

We evaluate performance when quantizing the transform coefficients in terms of Bjontegaard-based (BD) metrics. We use the BD-BR metric (%) for bit-rate and the BD-PSNR metric (dB) for reconstruction quality. The BD-BR metric represents the savings in bit-rate, where negative values mean that the transform uses fewer bits for the same video quality. The BD-PSNR metric represents the increase in video quality for the same number of bits used. We use the entropy [51] of the DCT coefficients as the lower limit for the average coding length in bits per pixel (bpp)

for 4 QPs, i.e., $QP = \{22, 27, 32, 37\}$. Table 6 tabulates the BD-PSNR and BD-BR values for all evaluated videos, while Table 7 summarizes performance by tabulating the average BD-PSNR and BD-BR values over all evaluated sequences. . The GBTs learned by the GBT-ONL framework provide BD-PSNR improvements of 1.01dB, 1.35dB, 2.15dB, 1.99 dB for Class A, B, C, and D sequences, respectively. The smallest improvements are attained for Class SC, due to the complexity of these sequences in terms of edges and textures. Specifically, the gains in video quality are only in the range $[0.08 - 0.20]$ dB for SC sequences.

The GBT-CNN and the GL-GBT attain stronger performance than that of the GBT-ONL framework, however, we should recall that those transforms need offline training. Compared to the DCT/DST, which do not require any training, the GBT-ONL framework attains the best performance and provides important improvements compared to the DCT. As expected, the KLT provides the best performance. However, as mentioned before, the KLT requires the storage of the covariance matrices in the compressed bit-stream, which makes it impractical.

In terms of the BD-BR, we can see important improvements introduced by the GBTs learned by the GBT-ONL framework. For the sequence *People_on_street* (Class A), the GBT-ONL framework saves 13.47% of bit-rate compared to the DCT. Similarly, for the sequence *BQ_mall* (Class C), the GBT-ONL framework needs 11.85% fewer bits than the DCT to store a video of similar quality. The performance on Class E sequences is particularly strong, as the GBT-ONL framework saves up to 14.88% of bit rate compared to the DCT. The performance on Class SC is comparatively low due to the presence of several edges in these sequences. As expected, the KLT outperforms all other transforms. Specifically, the KLT requires, on average, 70% fewer bits compared to the DCT to store the sequences at a similar quality. For example, for sequence *Traffic* (Class A), the KLT requires 78.56% fewer bits compared to DCT. It requires 76.65% and 70.07% less bits for sequences *Blowing_bubble* (Class E) and *Map* (Class SC).

The method proposed in [33] is similar to the proposed GBT-ONL framework in terms of learning GBTs online. However, the method in [33] requires signaling into the bitstream the usage of the GBT on each block. That method [33] attains a BD-BR value of -10.06, which is better than the BD-BR value of -9.20 attained by the GBT-ONL framework for all the sequences evaluated in this work. It is important to note that the BD-BR reported in [33] is that achieved by a complete compression pipeline, while that achieved by the GBT-ONL framework is for the block-wise transform process of a compression pipeline. Hence, the relatively better results achieved by the method in [33] in terms of BD-BR may be due to the use of a powerful entropy encoder to compress the transform coefficients.

Fig. 11 shows a reconstructed frame of the sequence *Basketball_drill* after transformation by several transforms and quantization with $QP = 37$. As depicted, the GBT-ONL framework achieves a higher visual quality than that achieved by the DCT. The GL-GBT, which requires offline training, achieves a visual quality very close to that achieved by the KLT.

4.5. Complexity and processing times

The complexity of GBT-ONL framework depends on the operations involved in the optimization processes to predict the graph Laplacian, the number of GD iterations, and the complexity of the eigen-decompensation. Predicting

the graph Laplacian matrix of a (square) residual block $\mathbf{S} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$ with N residual values using a shallow FC-NN has the following complexity:

$$O(N \times L \times O) \tag{5}$$

where N, L, and O are the number of neurons in the input, hidden layer, and output (i.e., the number of elements in the graph Laplacian), respectively. For P iterations of GD, the complexity is:

$$O(P \times N \times L \times O). \tag{6}$$

The eigendecomposition of the graph Laplacian $\mathbf{L} \in \mathbb{R}^{\sqrt{O}\times\sqrt{O}}$ with O elements has a complexity of:

$$O(O^3). \tag{7}$$

The total complexity of the GBT-ONL framework for a block with N residual values is then:

$$O(P \times N \times L \times O) + O(O^3). \tag{8}$$

Table 8, tabulates the average processing times, per frame, for each class of video sequence. As tabulated, the GBL-ONL framework has longer processing times due to the online optimization that has to be performed as blocks are processed. However, it offers several unique advantages. First, it enables dynamic adaptation to scene changes by learning a graph Laplacian for each residual block, ensuring that the transform is tailored to the local characteristics of the residual signal. Second, it achieves better data decorrelation, preserving more signal's energy in a small set of transform coefficients compared to the DCT. Third, it eliminates the need to signal any information into the compressed bitstream, since the same online optimization process can be replicated at the decoder side.

5. Conclusion

In this paper, we proposed the GBT-ONL framework to learn GBTs in the context of block-based PTC with intra-prediction. The GBT-ONL framework is based on online optimization of a shallow FC-NN designed to predict the graph Laplacian needed to compute the GBTs of a residual block. Since the optimization process uses only information available in the frame being processed, the GBT-ONL framework eliminates the need to signal extra information into the bitstream. Specifically, the same online optimization process can be replicated at the reconstruction stage. We evaluated the performance of the GBT-ONL framework in terms of the PE (%) and MSE when a small percentage of the largest coefficients are used for reconstruction, as well as in terms of the PSNR when different quantization levels are applied to the transform coefficients. We also evaluated performance in terms of the BD-PSNR and BD-BR metrics using the DCT as the reference transform. The evaluation results showed that the GBTs learned by the proposed GBT-ONL framework can outperform the DCT and the DCT/DST. These results also confirmed the advantages of

GBTs in adapting to new patterns by leveraging ML without the use of any offline training data or pre-trained models. Our future work includes two strategies to improve performance: 1) Extracting features from the imaging data by using convolutions. And 2) developing a hybrid framework that combines the strengths of GBT-ONL for edge-rich content with those of traditional transforms, e.g., DCT, for smooth content.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G. J. 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.

[2] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, R. Bultje, The latest open-source video codec vp9-an overview and preliminary results, in: 2013 Picture Coding Symposium (PCS), IEEE, 2013, pp. 390–393.

[3] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, et al., An overview of core coding tools in the av1 video codec, in: 2018 Picture Coding Symposium (PCS), IEEE, 2018, pp. 41–45.

[4] Versatile video coding, `https://bitmovin.com/vvc-video-codec/`, [Online].

[5] X. Zhao, J. Chen, M. Karczewicz, A. Said, V. Seregin, Joint separable and non-separable transforms for next-generation video coding, IEEE Transactions on Image Processing 27 (5) (2018) 2514–2525.

[6] Y.-H. Chao, H. E. Egilmez, A. Ortega, S. Yea, B. Lee, Edge adaptive graph-based transforms: Comparison of step/ramp edge models for video compression, in: 2016 IEEE International Conference on Image Processing (ICIP), IEEE, 2016, pp. 1539–1543.

[7] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Processing Magazine 30 (3) (2013) 83–98.

[8] A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs, IEEE Transactions on Signal Processing 61 (7) (2013) 1644–1656. doi:10.1109/TSP.2013.2238935.

[9] D. Roy, T. Guha, V. Sanchez, Graph-based transform with weighted self-loops for predictive transform coding based on template matching, in: 2019 Data Compression Conference (DCC), 2019, pp. 329–338.

[10] D. Roy, V. Sanchez, Graph-based transforms based on prediction inaccuracy modeling for pathology image coding, in: Data Compression Conference, 2018, pp. 157–166.

[11] E. Pavez, H. E. Egilmez, Y. Wang, A. Ortega, Gtt: Graph template transforms with applications to image coding, in: 2015 Picture Coding Symposium (PCS), IEEE, 2015, pp. 199–203.

[12] D. Roy, T. Guha, V. Sanchez, Graph based transforms based on graph neural networks for predictive transform coding, in: 2021 Data Compression Conference (DCC), 2021, pp. 367–367. doi:10.1109/DCC50243.2021.00079.

[13] D. Roy, T. Guha, V. Sanchez, Graph-based transform based on neural networks for intra-prediction of imaging data, in: 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP), 2021, pp. 1–6. doi:10.1109/MLSP52302.2021.9596317.

[14] B. Girault, E. Pavez, A. Ortega, Joint graph and vertex importance learning, in: 2023 31st European Signal Processing Conference (EUSIPCO), IEEE, 2023, pp. 1858–1862.

[15] M. Alawad, H. Yoon, S. Gao, B. Mumphrey, X. Wu, E. B. Durbin, J. Jeong, I. Hands, D. Rust, L. Coyle, L. Penberthy, G. Tourassi, Privacy-preserving deep learning nlp models for cancer registries, IEEE Transactions on Emerging Topics in Computing 9 (03) (2021) 1219–1230. doi:10.1109/TETC.2020.2983404.

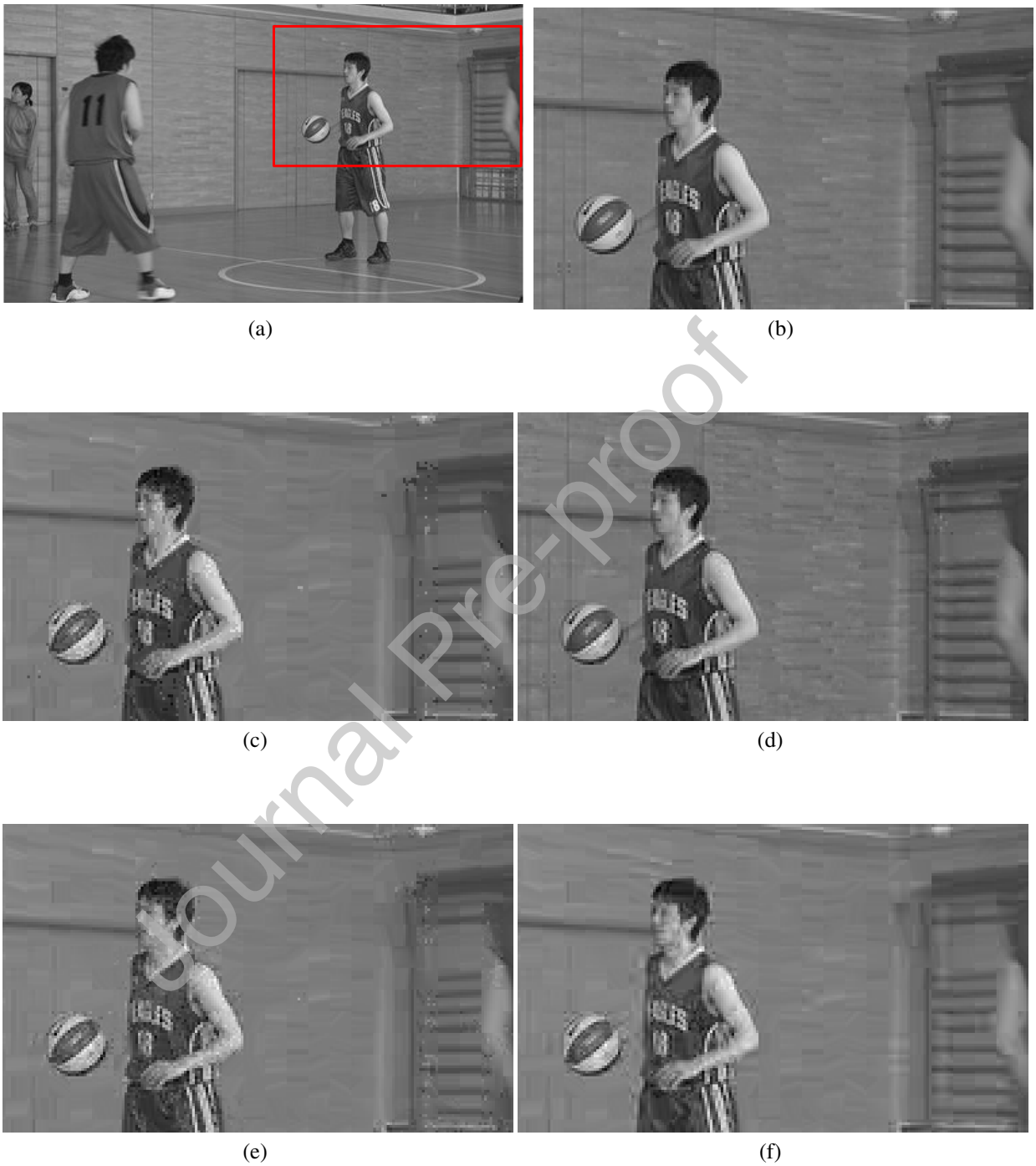


Figure 11. (a) An original frame of sequence *Basketball_pass* (Class D). (b) An area reconstructed after using the GL-GBT (PSNR = 31.3 dB), (c) the DCT (PSNR = 26.0 dB), (d) the KLT (PSNR = 28.7 dB), (e) the GBT-CNN (PSNR = 25.0 dB), and (f) the proposed GBT-ONL framework (PSNR = 26.6 dB). In all cases, $QP = 37$.

[16] J. Chen, T. Lange, M. Andjelkovic, A. Simevski, L. Lu, M. Krstic, Solar particle event and single event upset prediction from sram-based monitor and supervised machine learning, IEEE Transactions on Emerging Topics in Computing 10 (02) (2022) 564–580. doi:10.1109/TETC.2022.3147376.

[17] V. Sanchez, F. Aulí-Llinàs, J. Bartrina-Rapesta, J. Serra-Sagristà, Hevc-based lossless compression of whole slide pathology images, in: 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2014, pp. 297–301. doi:10.1109/GlobalSIP.2014.7032126.

[18] B. Elmeligy, T. Richter, R. R. R. Rao, S. Fößel, A. Raake, Evaluating visually lossless compression of jpeg xs, jpeg 2000, hevc and av1 in selected medical imaging modalities, in: 2024 16th International Conference on Quality of Multimedia Experience (QoMEX), IEEE, 2024, pp. 221–227.

[19] X. Liu, M. Wang, S. Wang, S. Kwong, Bilateral context modeling for residual coding in lossless 3d medical image compression, IEEE Transactions on Image Processing 33 (2024) 2502–2513.

[20] F. Li, O. Ieremeiev, V. Lukin, K. Egiazarian, Bpg-based lossy compression of three-channel remote sensing images with visual quality control, Remote Sensing 16 (15) (2024) 2740.

[21] P. S. Nonis, J. Schön, A. Nadobnik, T. Rambau, J. Jakobi, Performance differences between h. 264 and h. 265 video compression standards in the context of a remote tower optical system, in: 2025 Integrated Communications, Navigation and Surveillance Conference (ICNS), IEEE, 2025, pp. 1–8.

[22] H. E. Egilmez, A. Said, Y. H. Chao, A. Ortega, Graph-based transforms for inter predicted video coding, in: Image Processing, 2015 IEEE International Conference on, IEEE, 2015, pp. 3992–3996.

[23] A. Gnutti, F. Guerrini, R. Leonardi, A. Ortega, Coding of image intra prediction residuals using symmetric graphs, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 131–135.

[24] S. Bagheri, T. T. Do, G. Cheung, A. Ortega, Hybrid model-based / data-driven graph transform for image coding, in: 2022 IEEE International Conference on Image Processing (ICIP), 2022, pp. 3667–3671. doi:10.1109/ICIP46576.2022.9897653.

[25] C. Zhang, D. Florêncio, Analyzing the optimality of predictive transform coding using graph-based models, IEEE Signal Processing Letters 20 (1) (2012) 106–109.

[26] W. Hu, G. Cheung, A. Ortega, Intra-prediction and generalized graph fourier transform for image coding, IEEE Signal Processing Letters 22 (11) (2015) 1913–1917.

[27] A. Gnutti, F. Guerrini, R. Leonardi, A. Ortega, Variable-size symmetry-based graph fourier transforms for image compression, IEEE Transactions on Circuits and Systems for Video Technology.

[28] H. E. Egilmez, Y.-H. Chao, A. Ortega, Graph-based transforms for video coding, IEEE Transactions on Image Processing 29 (2020) 9330–9344.

[29] H. E. Egilmez, Y.-H. Chao, A. Ortega, B. Lee, S. Yea, Gbst: Separable transforms based on line graphs for predictive video coding, in: IEEE Int. Conf. Image Processing, IEEE, 2016, pp. 2375–2379.

[30] C.-Y. Zhang, Q. Zhao, C. P. Chen, W. Liu, Deep compression of probabilistic graphical networks, Pattern Recognition 96 (2019) 106979. doi:https://doi.org/10.1016/j.patcog.2019.106979.
URL https://www.sciencedirect.com/science/article/pii/S0031320319302821

[31] D. Roy, T. Guha, V. Sanchez, Graph-based transform based on 3d convolutional neural network for intra-prediction of imaging data, in: 2022 Data Compression Conference (DCC), 2022, pp. 212–221. doi:10.1109/DCC52660.2022.00029.

[32] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, Learning laplacian matrix in smooth graph signal representations, IEEE Transactions on Signal Processing 64 (23) (2016) 6160–6173. doi:10.1109/TSP.2016.2602809.

[33] W.-Y. Lu, E. Pavez, A. Ortega, X. Zhao, S. Liu, Adaptive online learning of separable path graph transforms for intra-prediction, in: 2024 Picture Coding Symposium (PCS), 2024, pp. 1–5. doi:10.1109/PCS60826.2024.10566291.

[34] W.-Y. Lu, E. Pavez, A. Ortega, X. Zhao, S. Liu, Online-learned graph transforms for adaptive block size intra-predictive coding, in: Applications of Digital Image Processing XLVII, Vol. 13137, SPIE, 2024, pp. 443–450.

[35] D. Recoskie, R. Mann, Learning filters for the 2d wavelet transform, in: 2018 15th Conference on Computer and Robot Vision (CRV), IEEE,

2018, pp. 198–205.

[36] G. Michau, G. Frusque, O. Fink, Fully learnable deep wavelet transform for unsupervised monitoring of high-frequency time series, Proceedings of the National Academy of Sciences 119 (8) (2022) e2106598119.

[37] Y. Sun, J. Chen, Q. Liu, G. Liu, Learning image compressed sensing with sub-pixel convolutional generative adversarial network, Pattern Recognition 98 (2020) 107051. doi:https://doi.org/10.1016/j.patcog.2019.107051.
URL https://www.sciencedirect.com/science/article/pii/S003132031930353X

[38] M. Sun, X. He, S. Xiong, C. Ren, X. Li, Reduction of jpeg compression artifacts based on dct coefficients prediction, Neurocomputing 384 (2020) 335–345. doi:https://doi.org/10.1016/j.neucom.2019.12.015.
URL https://www.sciencedirect.com/science/article/pii/S0925231219317175

[39] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, J. Yosinski, Faster neural networks straight from jpeg, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.
URL https://proceedings.neurips.cc/paper/2018/file/7af6266cc52234b5aa339b16695f7fc4-Paper.pdf

[40] A. Arrufat, P. Philippe, O. Déforges, Non-separable mode dependent transforms for intra coding in hevc, in: 2014 IEEE Visual Communications and Image Processing Conference, 2014, pp. 61–64. doi:10.1109/VCIP.2014.7051504.

[41] K. Fan, R. Wang, W. Lin, L.-Y. Duan, W. Gao, Signal-independent separable klt by offline training for video coding, IEEE Access 7 (2019) 33087–33093. doi:10.1109/ACCESS.2019.2903734.

[42] H. Abbas, M. Fahmy, A neural model for adaptive karhunen loeve transformation (klt), in: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, Vol. 2, 1992, pp. 975–980 vol.2. doi:10.1109/IJCNN.1992.226861.

[43] D. Pakiyarajah, E. Pavez, A. Ortega, D. Mukherjee, O. Guleryuz, K.-S. Lu, K. K. Sivakumar, Joint optimization of primary and secondary transforms using rate-distortion optimized transform design, arXiv preprint arXiv:2505.15104.

[44] Y. Ouyang, G. Shen, V. Sanchez, Look at adjacent frames: Video anomaly detection without offline training, in: European Conference on Computer Vision, Springer, 2022, pp. 642–658.

[45] R. Li, T. Ouyang, L. Zeng, G. Liao, Z. Zhou, X. Chen, Online optimization of dnn inference network utility in collaborative edge computing, IEEE/ACM Transactions on Networking 32 (5) (2024) 4414–4426. doi:10.1109/TNET.2024.3421356.

[46] Y. Gao, Y. Zhang, C. Yun, L. Huang, Online optimization of integrated energy systems based on deep learning predictive control, Electric Power Systems Research 243 (2025) 111510.

[47] SO/IEC-JCT1/SC29/WG11, Common Test Conditions and Software Reference Configurations, Geneva, Switzerland, 2012.

[48] H. E. Egilmez, Y. H. Chao, A. Ortega, Graph-based transforms for video coding, IEEE Transactions on Image Processing 29 (2020) 9330–9344. doi:10.1109/tip.2020.3026627.
URL http://dx.doi.org/10.1109/TIP.2020.3026627

[49] D. Roy, T. Guha, V. Sanchez, Graph-based transform based on 3d convolutional neural network for intra-prediction of imaging data, in: 2022 Data Compression Conference (DCC), IEEE, 2022, pp. 212–221.

[50] A. Saxena, F. C. Fernandes, Dct/dst-based transform coding for intra prediction in image/video coding, IEEE Transactions on Image Processing 22 (10) (2013) 3974–3981. doi:10.1109/TIP.2013.2265882.

[51] C. Thum, Measurement of the entropy of an image with application to image focusing, Optica Acta: International Journal of Optics 31 (2) (1984) 203–211. arXiv:https://doi.org/10.1080/713821475, doi:10.1080/713821475.
URL https://doi.org/10.1080/713821475

Table 2. PE (%) and MSE values using a small percentage of the largest coefficients. Results are for each sequence and on average per class and for all classes, including standard deviation values (SD ±).

Sequence	*Percentage of coefficients used for reconstruction*																							
	Baseline				Offline training								**Online training**				No training							
	KLT				GBT-CNN				GL-GBT				GBT-ONL (ours)				DCT				DCT/DST			
	PE↑		MSE ↓		PE ↑		MSE ↓		PE↑		MSE ↓		PE ↑		MSE ↓		PE ↑		MSE ↓		PE ↑		MSE ↓	
	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%	5%	10%
4:2:0 YUV sequences																								
Class A																								
Traffic	89.1	93.1	13.9	12.1	67.1	85.0	37.8	19.9	90.1	93.9	14.6	11.8	**67.1**	**83.5**	**38.0**	**20.4**	65.2	82.3	38.9	21.2	63.0	80.0	40.1	24.9
People_on_street	88.8	92.4	12.1	11.4	65.3	84.0	36.9	18.5	89.1	93.1	12.8	10.4	**65.2**	**83.7**	**37.1**	**19.1**	63.1	82.4	38.1	20.3	62.2	78.8	39.0	23.4
Nebuta_festival	87.2	90.9	13.6	10.8	63.8	82.8	33.6	16.5	87.4	90.2	11.4	10.5	**63.5**	**80.2**	**33.9**	**18.2**	63.2	78.5	34.4	20.1	58.5	75.1	39.3	21.4
Average	88.4	92.2	13.2	11.5	65.4	83.9	36.2	18.3	88.9	92.4	13.0	10.9	65.3	82.5	36.3	19.2	63.8	81.1	37.2	20.6	61.3	78.0	39.5	23.2
SD	1.02	1.12	0.96	0.65	1.65	1.10	2.21	1.70	1.36	1.94	1.47	2.96	1.80	1.97	2.15	1.11	1.18	2.22	2.40	0.59	2.40	2.55	0.59	1.76
Class B																								
Kimono	93.2	98.3	14.8	12.4	49.1	61.0	55.9	41.9	96.5	99.9	08.6	06.9	**43.5**	**60.5**	61.6	**46.2**	41.6	60.0	63.7	46.4	41.4	58.2	63.1	47.4
Cactus	92.6	97.4	13.1	12.2	48.5	60.2	55.1	41.2	96.	99.9	08.0	06.5	**42.4**	**59.8**	**60.7**	**44.4**	40.4	58.1	62.6	46.0	40.3	57.1	62.4	46.0
Park_scene	92.1	94.8	13.5	10.9	48.2	60.0	54.9	40.9	95.8	99.9	07.9	06.4	**42.7**	59.6	59.7	**43.7**	40.1	58.1	62.0	45.1	40.4	57.4	61.7	45.9
BQTerrace	93.0	95.4	11.6	10.5	49.0	60.1	55.0	41.0	96.0	99.9	08.0	06.7	**43.5**	**58.3**	**57.8**	**42.1**	41.5	56.4	59.9	44.2	39.9	56.3	61.9	44.8
Average	92.7	96.5	13.3	11.5	48.7	60.3	55.2	41.3	96.1	99.9	8.1	06.7	43.0	59.6	60.0	44.1	40.9	58.1	62.0	45.4	40.5	57.3	62.3	46.0
SD	0.49	1.65	1.32	0.94	0.42	0.46	0.46	0.45	0.30	0.00	0.32	0.22	0.56	0.92	1.63	1.70	0.76	1.47	1.60	0.98	0.64	0.79	0.62	1.07
Class C																								
Race_horse	92.0	95.9	09.5	07.9	52.9	70.3	46.8	35.6	93.9	94.5	06.1	06.0	48.1	**65.4**	**55.2**	**36.9**	46.3	63.2	57.0	39.1	46.3	62.2	58.2	42.2
BQMall	89.8	93.3	09.7	07.1	52.2	70.0	46.3	35.2	93.5	93.9	06.1	06.1	**48.0**	**64.5**	**53.9**	**37.1**	46.0	62.8	55.9	38.8	45.1	61.0	55.1	41.7
Party_scene	90.3	94.2	08.9	06.9	52.0	69.8	46.1	35.1	93.4	93.6	06.1	06.1	**47.2**	**64.1**	**52.8**	**36.2**	45.0	62.1	55.0	38.2	44.1	60.3	56.0	40.0
Basketball_drill	94.5	98.1	08.7	07.7	53.0	70.1	46.7	35.7	93.8	94.4	06.1	05.9	**47.5**	**63.3**	**49.4**	**37.3**	43.1	61.4	53.8	39.2	41.1	60.1	57.6	38.0
Average	91.6	95.4	09.1	07.4	52.5	70.1	46.5	35.4	93.7	94.1	06.1	06.0	47.7	64.3	52.8	36.9	45.1	62.4	55.4	38.8	44.2	60.9	56.7	40.5
SD	2.12	2.11	0.48	0.48	0.50	0.21	0.33	0.29	0.24	0.42	0.00	0.10	0.42	0.87	2.49	0.48	1.44	0.79	1.36	0.45	2.22	0.95	1.43	1.90
Class D																								
Race_horse_D	92.0	95.0	11.6	09.6	57.9	69.9	38.2	29.6	91.8	95.8	06.0	06.0	**53.4**	**69.1**	**48.6**	**33.3**	51.6	68.7	50.4	33.7	50.6	67.1	51.3	35.6
Blowing_bubble	90.7	94.5	10.0	08.4	58.4	70.3	38.8	30.0	92.1	97.1	06.2	06.2	**52.4**	**69.3**	**47.4**	**31.3**	50.2	67.4	49.6	33.2	49.5	66.2	50.8	34.3
BQ_square	90.2	94.2	09.8	08.1	58.0	70.1	38.5	29.9	92.0	95.9	06.1	06.1	**52.4**	**69.2**	49.3	**31.1**	50.1	67.3	**47.0**	33.0	49.1	66.0	50.1	34.1
Basketball_pass	89.8	94.0	09.3	07.6	57.8	70.0	38.3	29.7	92.0	94.6	06.0	06.0	**52.3**	**69.4**	**47.6**	**30.9**	50.5	67.9	49.4	32.4	49.2	66.1	51.4	34.7
Average	90.7	94.4	10.2	08.4	58.0	70.1	38.5	29.8	92.0	95.9	06.1	06.1	52.6	69.3	47.7	31.7	50.6	67.9	49.1	33.1	49.6	66.4	50.9	34.7
SD	0.96	0.43	0.99	0.85	0.26	0.17	0.26	0.18	0.13	1.02	0.10	0.10	0.52	0.13	0.89	1.11	0.69	0.64	1.47	0.54	0.69	0.51	0.59	0.67
Class E																								
Kristine_n_Sara	87.8	91.7	14.8	13.1	70.0	80.8	33.7	22.9	93.9	96.8	07.8	07.0	**60.3**	**78.1**	**40.6**	**23.3**	58.9	76.0	42.0	25.4	58.2	75.0	43.2	27.0
Four_people	87.6	91.5	48.4	13.0	69.3	79.9	33.1	21.9	93.1	95.9	07.3	06.5	**60.6**	**77.5**	**40.1**	**23.7**	58.8	75.9	41.9	25.3	58.1	74.6	43.1	26.9
Jhonny	88.5	91.9	15.8	13.6	69.3	80.1	33.6	21.8	93.4	96.1	07.9	06.7	**61.3**	**77.7**	**40.8**	**25.4**	59.4	75.9	42.7	27.2	58.7	75.7	43.7	27.3
Average	87.9	91.7	15.0	13.2	69.5	80.3	33.5	22.2	93.4	96.3	07.7	06.7	60.7	77.8	40.5	24.1	59.0	76.3	42.2	25.6	58.3	75.1	43.3	27.1
SD	0.47	0.20	19.12	0.32	0.40	0.47	0.32	0.61	0.40	0.47	0.32	0.25	0.51	0.31	0.36	1.12	0.32	0.06	0.44	1.07	0.32	0.56	0.32	0.21
Class SC																								
China_speed	89.0	92.4	14.0	12.1	55.1	66.1	44.2	31.9	92.7	96.9	06.0	04.8	48.5	**65.5**	**52.3**	**36.0**	46.4	63.6	54.4	37.9	**50.1**	63.1	55.3	39.0
Slide_show	88.3	91.9	13.5	11.4	54.3	65.0	43.0	30.9	92.2	96.2	05.8	04.3	48.1	**64.7**	**52.2**	**36.4**	46.1	63.4	54.2	37.7	**50.0**	62.9	55.0	38.9
Sc_Map	87.9	90.9	12.3	10.9	54.6	65.2	43.4	31.2	92.5	96.4	05.9	04.5	47.7	**65.1**	**52.2**	**35.4**	45.9	63.2	54.0	37.3	**49.9**	62.7	54.8	38.5
Sc_Prog	87.2	92.2	12.6	11.1	54.6	64.9	43.3	31.3	92.9	96.7	06.0	04.7	47.6	**64.1**	**51.9**	**36.1**	45.8	63.1	53.7	37.1	**49.8**	62.2	54.7	38.5
Average	88.1	91.9	13.1	11.3	54.7	65.3	43.5	31.3	92.6	96.5	05.9	04.6	48.0	64.9	52.2	36.0	46.1	63.8	54.1	37.5	50.0	62.7	55.0	37.3
SD	0.75	0.67	0.79	0.53	0.33	0.55	0.51	0.42	0.30	0.31	0.10	0.22	0.41	0.60	0.17	0.42	0.26	0.22	0.30	0.37	0.13	0.39	0.26	0.26
Overall avg.	89.7	93.4	12.2	10.4	60.0	73.2	39.6	28.1	92.4	95.9	07.7	06.8	**54.3**	**71.2**	**46.6**	**30.3**	52.4	69.6	48.5	31.9	51.9	68.1	49.8	33.5
Overall SD	0.61	0.74	7.44	0.23	0.52	0.33	0.75	0.56	0.45	0.70	0.60	0.26	0.54	0.65	0.96	0.48	0.47	0.82	0.78	0.29	0.99	0.81	0.42	0.73

Notes: The **bold values denote the best results, while *underlined* values indicate the second-best results within the respective comparisons.

Table 3. Average PE (%) and MSE values using a small percentage of the largest coefficients.

	Transform	PE ↑		MSE ↓	
		5%	10%	5%	10%
Baseline	KLT	89.7	93.4	12.2	10.4
Offline training	GBT-CNN	60.0	73.2	39.6	28.1
	GL-GBT	92.4	95.9	07.7	06.8
Online training	GBT-ONL	54.3	71.2	46.6	30.3
No training	DCT	52.4	69.6	48.5	31.9
	DCT/DST	51.9	68.1	49.8	33.5

Table 4. PSNR (dB) values when using quantization on the transform coefficients. Results are for each sequence and on average per class and for all classes, including standard deviation values (SD ±)

	Quantization parameters																							
Sequence	Baseline				Offline training								Online training				No training							
	KLT				GBT-CNN				GL-GBT				GBT-ONL (ours)				DCT				DCT/DST			
QP	22	27	32	37	22	27	32	37	22	27	32	37	22	27	32	37	22	27	32	37	22	27	32	37
Class A																								
Traffic	37.5	33.2	29.8	27.8	33.3	30.3	29.4	25.2	34.0	33.0	30.6	22.1	**33.4**	30.9	**28.5**	**23.3**	33.3	**31.3**	27.5	21.2	21.0	20.1	19.5	16.1
People_on_street	38.7	34.4	31.0	29.0	34.5	31.5	30.6	26.4	35.2	34.2	31.8	23.3	**34.6**	32.2	**29.7**	**24.4**	34.5	**32.5**	28.7	22.4	22.0	19.3	18.6	17.3
Nebuta_festival	39.5	35.2	31.8	29.8	35.3	32.3	31.4	27.2	36.0	35.0	32.6	24.1	**35.4**	32.9	**30.4**	**25.3**	35.3	**33.3**	29.5	23.2	21.7	20.2	19.5	17.0
Average	38.6	34.3	30.9	28.9	34.4	31.4	30.5	26.3	35.1	34.1	31.7	23.2	34.5	32.0	29.5	24.3	34.4	32.4	28.6	22.3	21.6	19.9	19.2	16.8
SD	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.01	0.96	1.00	1.01	1.01	1.01	1.01	0.51	0.49	0.52	0.62
Class B																								
Kimono	37.4	35.4	30.9	29.4	30.5	28.4	26.4	22.2	34.5	32.6	28.4	24.5	31.7	29.8	27.4	**22.0**	**32.8**	**30.9**	**28.1**	21.5	19.2	16.9	16.2	15.4
Cactus	38.1	36.1	31.7	30.0	31.2	29.0	27.	22.9	35.1	33.3	29.2	25.3	32.1	30.2	27.8	**22.6**	**32.9**	**31.1**	**28.3**	22.1	20.3	17.9	16.9	16.5
Park_scene	36.0	34.0	29.5	28.0	29.1	27.0	25.0	20.8	33.1	31.2	27.0	23.1	30.5	28.6	26.3	**20.6**	**31.4**	**29.9**	**27.1**	20.1	20.1	19.7	18.6	15.5
BQTerrace	38.8	36.7	32.3	30.7	35.1	31.7	30.5	23.5	35.8	33.4	29.8	25.9	**34.1**	**31.6**	**29.5**	**23.3**	32.9	31.1	28.4	22.8	21.6	20.9	20.4	17.7
Average	37.6	35.6	31.1	29.5	31.5	29.0	27.2	22.4	34.6	32.6	28.6	24.7	32.1	30.1	27.8	22.1	32.5	30.8	28.0	21.6	20.3	18.9	18.0	16.3
SD	1.20	1.16	1.21	1.15	2.57	1.97	2.34	1.16	1.15	1.01	1.21	1.21	1.50	1.24	1.33	1.15	0.73	0.57	0.60	1.15	0.99	1.79	1.88	1.07
Class C																								
Race_horse	38.8	36.0	30.7	23.8	30.8	28.6	26.7	22.5	34.8	32.9	28.7	24.8	32.2	29.5	27.7	**22.3**	**33.1**	**30.0**	**28.1**	21.8	20.5	16.8	16.2	15.6
BQMall	39.2	36.4	31.1	24.3	31.2	29.2	27.1	23.0	35.2	33.3	29.2	25.3	32.5	29.8	27.8	**22.8**	**33.5**	**30.0**	**28.1**	22.2	19.9	19.0	17.7	15.9
Party_scene	37.1	34.3	29.0	22.1	29.1	27.0	25.0	20.8	33.1	31.2	27.0	23.1	30.6	28.6	26.4	**20.6**	**31.4**	**29.9**	**27.1**	20.1	17.9	15.8	15.1	14.0
Basketball_drill	39.7	36.9	31.7	24.8	34.6	31.7	28.7	23.5	35.7	33.8	29.7	25.9	**34.3**	**31.1**	**28.7**	**23.3**	33.5	30.1	28.2	22.8	20.9	19.9	18.0	17.7
Average	38.7	35.9	30.6	23.8	31.4	29.1	26.9	22.5	34.7	32.8	28.7	24.8	32.4	29.8	27.7	22.3	32.9	30.0	27.9	21.7	19.8	17.9	16.8	15.8
SD	1.13	1.13	1.16	1.17	2.30	1.95	1.52	1.17	1.13	1.13	1.17	1.20	1.52	1.03	0.95	1.17	1.00	0.08	0.52	1.16	1.33	1.90	1.35	1.52
Class D																								
Race_horse_D	40.5	38.4	34.1	27.0	34.5	33.1	30.0	23.4	39.7	37.9	32.6	29.7	**34.5**	**32.5**	**29.0**	24.0	34.5	31.6	28.0	**24.4**	22.2	21.4	20.1	19.3
Blowing_bubble	41.9	40.0	35.4	28.4	37.9	35.6	32.4	24.7	41.1	39.2	34.0	31.1	**37.0**	**34.2**	**30.8**	25.3	35.9	23.6	29.0	**25.7**	23.6	22.4	21.1	20.6
BQ_square	39.1	37.0	32.7	25.6	33.1	31.7	28.6	22.0	38.3	36.5	31.2	28.3	**33.2**	**31.6**	**27.9**	22.6	33.1	31.2	27.0	**23.0**	20.8	20.0	19.0	17.9
Basketball_pass	42.2	40.3	35.7	28.7	37.9	36.8	32.7	25.0	41.4	39.5	34.3	31.3	**37.1**	**35.0**	**31.1**	25.6	36.0	32.9	29.3	**26.0**	23.7	22.7	21.3	20.9
Average	40.9	38.9	34.5	27.4	35.9	34.3	30.9	23.8	40.1	38.3	33.0	30.1	35.5	33.3	29.7	24.4	34.9	29.8	28.3	24.8	22.6	21.6	20.4	19.7
SD	1.42	1.53	1.37	1.42	2.44	2.32	1.97	1.37	1.42	1.37	1.42	1.40	1.92	1.55	1.52	1.37	1.37	4.21	1.04	1.37	1.37	1.22	1.06	1.37
Class E																								
Kristine_and_Sara	40.6	39.3	35.5	30.7	36.4	34.6	31.9	25.0	36.7	33.7	31.3	26.7	**36.4**	**34.5**	**31.5**	25.8	36.3	34.0	31.2	**26.3**	24.0	22.3	21.2	21.1
Four_people	39.0	37.7	33.9	29.1	34.8	33.0	30.3	23.4	35.1	32.1	29.7	25.1	**34.8**	**33.2**	**30.0**	24.2	34.7	33.0	29.6	**24.7**	22.4	21.8	20.9	19.6
Jhonny	41.8	40.5	36.7	31.9	37.6	35.8	33.1	26.3	38.0	34.9	32.5	28.0	**37.6**	**35.6**	**32.6**	27.0	37.5	35.0	32.2	**27.5**	24.0	21.3	20.2	17.7
Average	40.5	39.2	35.4	30.6	36.3	34.5	31.8	24.9	36.6	33.6	31.2	26.6	36.3	34.4	31.4	25.7	36.2	34.0	31.0	26.2	23.5	21.8	20.8	19.5
SD	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.45	1.45	1.40	1.40	1.45	1.40	1.20	1.31	1.40	1.40	1.00	1.31	1.40	0.92	0.50	0.51	1.70
Class SC																								
China_speed	40.2	37.0	33.9	33.1	36.1	33.7	27.4	24.6	36.9	34.0	30.5	25.1	**35.8**	**33.4**	28.8	25.4	34.9	32.8	**29.6**	**26.0**	20.4	18.7	18.1	17.9
Slide_show	42.1	35.1	32.0	31.2	34.2	31.8	25.5	22.7	35.0	32.1	28.6	23.2	**34.4**	32.1	27.6	23.4	33.9	**32.3**	**29.1**	**24.1**	20.4	18.4	18.1	18.0
Sc_Map	40.9	37.6	34.6	33.8	36.7	34.3	28.1	25.2	37.5	34.7	31.2	25.8	**36.3**	**33.7**	29.4	26.1	35.5	32.8	**30.3**	**26.7**	23.0	19.4	18.8	18.6
Sc_Programming	41.2	37.9	34.9	34.1	40.1	36.6	31.4	25.5	37.9	35.0	31.6	26.1	**38.0**	**34.9**	**30.9**	26.4	35.6	32.8	30.3	**27.0**	23.3	22.7	22.3	21.9
Average	41.1	36.9	33.9	33.1	36.8	34.1	28.1	24.5	36.8	34.0	30.5	25.1	36.1	33.5	29.2	25.3	35.0	32.7	29.8	26.0	21.8	19.8	19.3	19.1
SD	0.79	1.26	1.30	1.30	2.46	1.98	2.46	1.26	1.28	1.30	1.33	1.30	1.49	1.15	1.37	1.35	0.78	0.25	0.59	1.30	1.59	1.98	2.01	1.89
Overall avg.	39.6	36.8	32.7	28.8	34.3	32.0	29.1	23.9	36.4	34.3	30.5	25.8	**34.4**	**32.1**	**29.1**	**23.9**	34.2	31.9	28.8	23.7	20.6	19.0	18.2	17.0
Overall SD	0.24	0.19	0.15	0.16	0.66	0.48	0.57	0.16	0.18	0.18	0.16	0.16	0.29	0.19	0.23	0.16	0.28	1.53	0.32	0.15	0.39	0.69	0.65	0.46

Notes: The **bold values denote the best results, while *underlined* values indicate the second-best results within the respective comparisons.

Table 5. Average PSNR (dB) values when using quantization on the transform coefficients.

	Transform	QP			
		22	27	32	37
Baseline	KLT	39.6	36.8	32.7	28.8
Offline Training	GBT-CNN	34.3	32.0	29.1	23.9
	GL-GBT	36.4	34.3	30.5	25.8
Online Training	GBT-ONL	34.4	32.1	29.1	23.9
No training	DCT	34.2	31.9	28.8	23.7
	DCT/DST	20.6	19.0	18.2	17.0

Table 6. BD-PSNR, BD-BR values (with respect to the DCT) when using quantization on the transform coefficients. Results are for each sequence and on average per class and for all classes, including standard deviation values (SD ±).

	Methods									
Sequence	*Baseline*		*Offline training*				**Online training**		*No training*	
	KLT		GBT-CNN		GL-GBT		GBT-ONL (ours)		DCT/DST	
	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR
					Class A					
Traffic	08.16	-78.56	01.17	-14.87	4.69	-31.44	**01.01**	**-11.63**	-6.39	23.62
People_on_street	07.02	-80.64	00.74	-15.61	3.83	-33.14	**00.59**	**-13.47**	-7.01	21.11
Nebut_festival	07.29	-80.35	00.92	-15.02	3.96	-33.04	**00.81**	**-12.23**	-6.32	22.94
Average	7.5	-79.9	0.9	-15.2	4.2	-32.5	0.8	-12.4	-6.6	22.6
SD	0.60	1.13	0.22	0.39	0.46	0.95	0.21	0.94	0.38	1.30
					Class B					
Kimono	03.75	-67.21	0.45	-09.23	02.33	-37.76	**00.31**	**-06.44**	-4.11	19.47
Cactus	02.91	-68.23	0.67	-08.22	1.31	-38.69	**00.51**	**-06.08**	-4.54	19.93
Park_scene	03.92	-67.30	1.03	-13.22	02.24	-37.67	**00.90**	**-11.08**	-5.12	16.94
BQTerrace	03.06	-68.18	1.51	-14.79	1.36	-38.64	**1.35**	**-11.55**	-3.82	13.01
Average	3.4	-67.7	0.9	-11.4	1.8	-38.2	0.8	-8.8	-4.4	17.3
SD	0.50	0.55	0.46	3.14	0.55	0.55	0.46	2.93	0.57	3.17
					Class C					
Race_horse	10.83	-52.33	01.11	-12.43	5.17	-46.19	**00.97**	**-09.44**	-09.16	30.01
BQMall	09.99	-53.26	00.94	-14.99	4.15	-47.12	**00.78**	**-11.85**	-08.29	27.81
Party_scene	08.01	-52.24	02.26	-11.56	05.08	-46.10	**02.15**	**-08.95**	-06.77	20.76
Basketball_drill	10.13	-53.21	01.46	-12.03	04.20	-47.07	**01.30**	**-09.29**	-07.77	21.45
Average	9.7	-52.8	1.4	-12.8	4.7	-46.6	1.3	-9.9	-8.0	25.0
SD	1.21	0.55	0.59	1.53	0.55	0.55	0.61	1.33	1.00	4.60
					Class D					
Race_horse_D	10.67	-75.72	02.12	-09.32	08.50	-75.26	**01.98**	**-06.53**	-15.27	32.85
Blowing_bubble	09.65	-76.65	02.15	-09.12	-07.66	-76.28	**01.99**	**-06.98**	-14.04	31.51
BQ_square	10.58	-75.63	01.06	-12.94	08.67	-75.35	**00.95**	**-10.80**	-11.88	29.03
Basketball_pass	09.70	-76.60	01.65	-11.22	07.81	-76.23	**01.49**	**-07.98**	-14.96	32.59
Average	10.2	-76.2	1.7	-10.7	4.3	-75.8	1.6	-8.1	-14.0	31.5
SD	0.55	0.55	0.51	1.80	8.00	0.55	0.49	1.92	1.53	1.74
					Class E					
Kristine_and_Sara	08.47	-69.89	0.56	-17.34	02.52	-22.28	**00.40**	**-14.10**	-12.92	35.96
Four_people	07.33	-71.97	0.64	-17.02	01.66	-23.98	**00.49**	**-14.88**	-11.93	36.01
Jhonny	07.60	-71.68	1.01	-10.50	01.79	-23.88	**00.90**	**-07.71**	-10.72	19.34
Average	7.8	-71.2	0.7	-15.0	2.0	-23.4	0.6	-12.2	-11.9	30.4
SD	0.60	1.13	0.24	3.86	0.46	0.95	0.27	3.93	1.10	9.61
					Class SC					
China_speed	09.21	-69.05	0.33	-08.61	06.15	-40.27	**00.19**	**-05.82**	-14.09	35.90
Slide_show	08.52	-70.02	0.36	-07.92	05.18	-41.16	**00.20**	**-05.78**	-14.71	35.01
Sc_Map	08.37	-70.07	0.21	-06.22	05.13	-41.20	**00.08**	**-04.08**	-15.82	37.90
Sc_Programming	10.21	-69.14	0.34	-09.05	06.06	-40.19	**00.18**	**-05.81**	-14.28	36.03
Average	9.1	-69.6	0.3	-8.0	5.6	-40.7	0.2	-5.4	-14.7	36.2
SD	0.84	0.55	0.07	1.24	0.55	0.55	0.06	0.86	0.77	1.21
Overall average	7.97	-69.0	1.03	-11.87	3.82	-44.22	**0.89**	**-9.20**	-10.0	27.24
Overall SD	0.27	0.30	0.20	1.28	3.06	0.21	0.21	1.22	0.41	3.22

Notes: The **bold values denote the best results, while *underlined* values indicate the second-best results within the respective comparisons.

Table 7. Average BD-PSNR and BD-BR values (with respect to DCT) when using quantization on the transform coefficients.

	Transform	Metric	
		BD-PSNR	BD-BR
Baseline	KLT	7.97	-69.0
Offline Training	GBT-CNN	1.03	-11.87
	GL-GBT	3.82	-44.22
Online Training	GBT-ONL	0.89	-9.20
No training	DCT/DST	-10.0	27.24

Table 8. Average processing times, per frame, in seconds(s) and milliseconds (ms) for several transforms.

Video	**Baseline**	**Offline training**				**Online training**	**No training**	
	KLT	GBT-CNN		GL-GBT		GBT-ONL (ours)	DCT	DCT/DST
	(ms)	Train (s)	Test (s)	Train (s)	Test(ms)	(s)	(ms)	(ms)
Class A								
Traffic	102.31	500.32	0.45	654.34	93.21	1.47	12.42	22.14
People_on_street	75.76	598.43	0.62	634.23	105.32	2.63	17.23	25.90
Nebuta_festival	119.22	845.12	0.83	900.34	105.34	4.21	20.54	30.67
Class B								
Kimono	65.12	734.45	0.32	830.12	72.46	4.12	15.48	22.14
Cactus	54.09	890.67	0.75	940.00	69.22	7.34	13.82	21.98
Park_scene	69.12	934.75	0.91	993.20	79.31	10.25	17.45	26.21
BQTerrace	45.17	857.27	0.34	860.30	58.34	5.96	16.23	21.72
Class C								
Race_horse	30.48	327.67	0.58	412.93	39.12	2.65	16.47	17.15
BQMall	38.87	333.60	0.78	350.49	46.28	5.12	20.16	26.63
Party_scene6	34.54	379.34	0.62	438.45	39.27	4.56	18.38	23.56
Basketball_drill	31.73	365.03	0.51	412.32	35.88	3.99	17.44	20.61
Class D								
Race_horse_D	20.89	280.11	0.43	300.45	25.49	2.67	23.66	27.39
Blowing_bubble	17.44	162.82	0.49	183.89	25.41	1.98	21.79	29.33
BQ_square	38.72	218.44	0.70	249.46	47.29	4.32	13.76	17.89
Basketball_pass	36.93	284.93	0.74	319.50	40.71	3.83	12.11	16.74
Class E								
Kristine_and_Sara	61.86	482.17	0.65	543.96	72.11	3.33	39.23	49.22
Four_people	65.90	413.91	0.71	458.45	69.25	4.21	47.35	54.23
Jhonny	87.48	517.33	0.82	562.45	95.56	7.11	51.26	62.01
Class SC								
China_speed	149.34	615.15	1.21	648.67	200.40	6.91	80.12	120.05
Slide_show	138.41	522.43	1.10	579.04	183.56	5.62	73.67	104.38
Sc_Map	157.18	631.54	1.26	735.60	190.80	7.01	86.09	143.29
Sc_Programming	146.95	421.55	1.25	502.56	160.32	6.21	82.71	119.72