Graph-based Transform based on 3D Convolutional Neural Network for Intra-Prediction of Imaging Data

Debaleena Roy*, Tanaya Guha†, and Victor Sanchez*

* Department of Computer Science
University of Warwick, UK
Coventry, UK

†School of Computing Science
University of Glasgow
Glasgow, UK

* {debaleena.roy,v.f.sanchez-silva}@warwick.ac.uk

Abstract

This paper presents a novel class of Graph-based Transform based on 3D convolutional neural networks (GBT-CNN) within the context of block-based predictive transform coding of imaging data. The proposed GBT-CNN uses a 3D convolutional neural network (3D-CNN) to predict the graph information needed to compute the transform and its inverse, thus reducing the signalling cost to reconstruct the data after transformation. The GBT-CNN outperforms the DCT and DCT/DST, which are commonly employed in current video codecs, in terms of the percentage of energy preserved by a subset of transform coefficients, the mean squared error of the reconstructed data, and the transform coding gain according to evaluations on several video frames and medical images.

1. Introduction

In block-based predictive transform coding (PTC), a video frame is divided into several non-overlapping blocks (see Fig. 1 (a)) and a residual block for each block is obtained by computing the difference between the original and predicted blocks. Each residual block is then transformed and the resulting transform coefficients are quantized and encoded to create a compressed bit-stream. Intra-prediction is a popular method to predict a block using reconstructed neighboring blocks of the same frame. For example, the High Efficiency Video Coding (HEVC) standard uses 33 angular prediction modes that model 33 different directional patterns; a DC mode and a Planar mode that generate smooth surfaces (see Fig. 1 (b)-(c)), while the Versatile Video Coding (VVC) standard uses 65 different directional modes for prediction.

Within block-based PTC, the transform plays a fundamental role to regulate coding performance by generating decorrelated coefficients that can compact the energy of the signal into a few significant transform coefficients. It is widely known that the Karhunen Loève Transform (KLT) is the linear transform with the best energy compaction properties for any arbitrary signal with a known covariance matrix, since it can represent most of the signal energy with only a few transform coefficients. The KLT performs the eigendecomposition of the covariance matrix of the signal. Because the KLT basis functions of natural images are similar to those of the Discrete Cosine Transform (DCT), the DCT has been championed as the best transform for compression.

Recently, the Graph-Based Transform (GBT) has shown promising results for data decorrelation and energy compaction in block-based PTC [1, 2, 3]. This is due

to the fact that the GBT is very adaptable to the signal since a unique graph is produced for each residual block to correctly depict the intrinsic structure and the correlation among the residual values [4, 5, 6, 7, 8, 9]. In [5, 6], we show that in terms of energy compaction and reconstruction quality, the GBT outperforms the DCT and the combination of DCT/DST (Discrete Sine Transform), as applied in modern video codecs. By leveraging a priori knowledge about signals represented by a graph template, Pavez *et al.* [7] propose the Graph Template Transform (GTT) to approximate the KLT. Egilmez *et al.* present a GBT based on Gaussian-Markov Random Field (GMRF) models that learn graphs from data and provide optimum separable and non-separable GBT solutions, called the GL-GBT [8]. To obtain an optimal GL-GBT, they solve an optimization problem with graph connectivity constraints, a data-fidelity term and a log-determinant Bregman divergence [9]. This optimization corresponds to the maximum likelihood estimation of the inverse covariance (precision) matrices for multivariate Gaussian distributions.

When the GBT is used in block-based PTC, the graph used to compute the GBT of each block at the encoder should be available to compute the inverse GBT during reconstruction at the decoder. This additional data should then be signalled into the bitstream, increasing the overhead. Our previous works in [10, 11] show an attractive solution for learning a mapping function to design a GBT without requiring to signal additional information. To make the same graph available to the decoder for reconstruction, a template-based prediction strategy is used to predict the residual followed by a neural network (NN) that estimates the graph to perform the GBT. Specifically, this approach involves two prediction methods: predicting the residuals and using the predicted residuals as an input to the NN to predict the graph. This approach, unfortunately, tends to degrade the quality of the reconstructed residual at the decoder. To address this issue, this paper introduces a novel class of GBT based on a 3D CNN (GBT-CNN), which uses the 3 reconstructed blocks surrounding the block to be encoded as input. We use a 3D CNN because the 3D convolution allows exploiting the relationship between these surrounding blocks, which are expected to be similar to the one to be encoded, by treating them as a single volume. These features are used to predict the adjacency matrix of the block to be encoded. More specifically, our proposed method maps these 3 surrounding blocks in the pixel domain to the graph representing the residual block to be encoded by using an encoding-decoding architecture. These 3 surrounding blocks are used to compute the same graph at the decoder, thus allowing to perform the inverse GBT. Our approach then avoids signalling extra information into the bitstream. To the best of our knowledge, no approach for learning a graph using 3D CNNs within the context of block-based PTC and GBTs has been proposed before. In terms of percentage of preserved energy (PE), mean squared error (MSE), Peak Signal-to-Noise Ratio (PSNR), and the transform coding gain, our evaluations on several video frames and medical images show that the proposed GBT-CNN outperforms the DCT/DST, DCT, and other similar GBTs [5, 6, 10, 11].

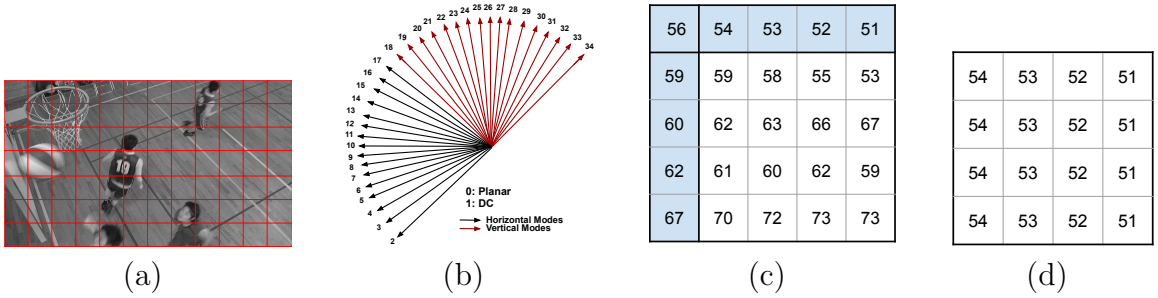

Figure 1: (a) Partition of a frame into non-overlapping blocks. (b) HEVC intra-prediction directions. (c) Sample block with reference samples used for prediction shaded in blue. (d) Block predicted by the mode 26 for the block in (c).

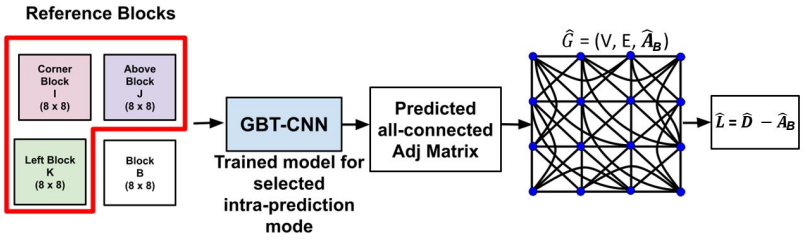

Figure 2: GBT-CNN used to predict matrix $\mathbf{A}$ for the current residual block. In this work, we use an all-connected topology for the graphs.

2. Proposed GBT-CNN

Let us denote a (square) residual block as $\mathbf{S} \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}}$, with a total of N residual values. $\mathbf{S}$ can be represented as an undirected weighted graph, $G = (V, E, \mathbf{A})$, where $V = \{v_n\}_{n=1}^{N}$ is the set of N nodes, E is the set of edges, and $\mathbf{A} \in \mathbb{R}^{N\times N}$ is the normalized symmetric adjacency matrix. The matrix $\mathbf{A}$ of a weighted graph stores the edge weights. The GBT for $\mathbf{S}$ can be computed by the eigendecomposition of the graph Laplacian, $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. The eigendecomposition of $\mathbf{L}$ can be used as an orthogonal transform for $\mathbf{S}$, since it has a complete set of eigenvectors with real, non-negative eigenvalues [12].

As the graph Laplacian requires the computation of the matrix $\mathbf{A}$, our objective is to develop a mapping between the 3 reconstructed blocks surrounding the block to be encoded and the matrix $\mathbf{A}$ of the residual block to be encoded. To this end, we aim to learn a mapping function of the form:

$$\mathbf{A_B} \approx f(\mathbf{B_{[I,J,K]}}), \tag{1}$$

where $\mathbf{B_{[I,J,K]}}$ represents a matrix with the 3 reconstructed gray scale blocks surrounding the block $\mathbf{B}$ to be encoded and $\mathbf{A_B}$ is the adjacency matrix of the graph of its residual block (see Fig. 2). Our solution to learn the mapping function in Eq. 1 is based on an encoding-decoding 3D CNN, as depicted in Fig. 3 for the case of 8×8 blocks. In this architecture, the convolution takes place over 3 layers of the encoder to extract feature maps $\mathbf{Z}^{(l_e)}$ where $\mathbf{Z}^{(0)} = \mathbf{B_{[I,J,K]}}$ is the input and $l_e \in [1, 3]$ denotes the layer number. After the convolutional layers, the feature maps are vectorized as an input to the decoder part of the architecture. Specifically, $\mathbf{Z}^{(l_e=3)}$ is transformed back to a reconstructed vector $\hat{\mathbf{a}}_{u,\mathbf{B}}$ by the decoder over a number of fully-connected

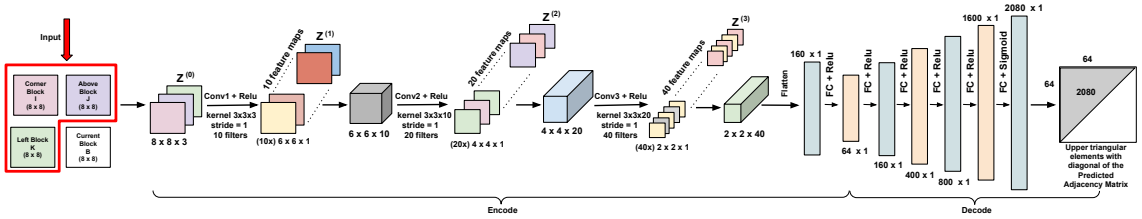

Figure 3: Architecture of the proposed GBT-CNN for 8×8 blocks.

(FC) layers:

$$\hat{\mathbf{a}}_{u,\mathbf{B}} = h(\mathbf{W}^{(l_d)}\mathbf{Z}^{(l_d-1)}), \tag{2}$$

where $\hat{\mathbf{a}}_{u,\mathbf{B}}$ is the prediction of the vectorized upper triangular matrix of $\mathbf{A_B}$, $h(\cdot)$ denotes an activation function, $\mathbf{W}^{(l_d)}$ is a weight matrix for the decoder layer l_d, and $\mathbf{Z}^{(l_d-1)}$ is the hidden representation produced by the decoder layer $(l_d - 1)$. For each FC layer, we apply the *ReLu* activation function, while the *Sigmoid* activation function is applied to the last layer of the encoder. The decoder consists of 6 FC layers. Note that the network only predicts the upper triangular elements and the diagonal of matrix $\mathbf{A_B}$. To obtain a complete predicted matrix $\hat{\mathbf{A}}_\mathbf{B}$, we mirror the elements of the upper diagonal to the lower diagonal:

$$\hat{\mathbf{A}}_\mathbf{B} = \hat{\mathbf{A}}_{u,\mathbf{B}} + (\hat{\mathbf{A}}_{u,\mathbf{B}})^T - Diag(\hat{\mathbf{A}}_{u,\mathbf{B}}), \tag{3}$$

where $\hat{\mathbf{A}}_{u,\mathbf{B}}$ is the matrix form of $\hat{\mathbf{a}}_{u,\mathbf{B}}$, $Diag(\hat{\mathbf{A}}_{u,\mathbf{B}})$ is the diagonal elements of $\hat{\mathbf{A}}_{u,\mathbf{B}}$ and $(\hat{\mathbf{A}}_{u,\mathbf{B}})^T - Diag(\hat{\mathbf{A}}_{u,\mathbf{B}})$ denotes the lower triangular matrix. We optimize the GBT-CNN by minimizing the following loss function:

$$L = \| \hat{\mathbf{a}}_\mathbf{B} - \mathbf{a_B} \|_2^2 + \lambda \| \mathbf{W}(:) \|_2, \tag{4}$$

where $\hat{\mathbf{a}}_\mathbf{B}$ is the complete predicted matrix $\hat{\mathbf{A}}_{u,\mathbf{B}}$ (see Eq. 3) in vectorized form, $\mathbf{a_B}$ is the vectorized form of the ground truth matrix $\mathbf{A_B}$, $\| \cdot \|$ is the L_2 norm, $\mathbf{W}(:)$ represents the learnable parameters in vector form, and λ controls the amount of L_2 regularization on the learnable parameters. The graph used to compute the GBT for the current residual block is then $\hat{G} = (V, E, \hat{\mathbf{A}}_\mathbf{B})$. To reconstruct the current block, the same graph used to compute the GBT should be used to compute the inverse GBT at the decoder. To this end, the same reconstructed blocks used as input are available at the decoder to predict matrix $\mathbf{A_B}$ by the trained GBT-CNN. Fig. 4 illustrates the complete compression framework assuming the trained GBT-CNN is common knowledge between encoder and decoder. As a result, our solution does not require signalling any additional data in the compressed bit-stream.

3. Performance Evaluation

Based on the 35 HEVC intra-prediction modes, we train 5 distinct networks: one for horizontal (H) modes (modes 3 - 17), one for vertical (V) modes (modes 9 - 13), one for diagonal (D) modes (modes 2, 18 and 34), one for the DC mode, and one for the planar (P) mode (see Fig. 2 and Fig. 5). We use 8×8 blocks and graphs with unit edge (UE) weights and an all-connected (All-C) topology with no self-loops. We use 40 distinct grey level YUV frames from Class A, B, C, D, E, and Screen Content,

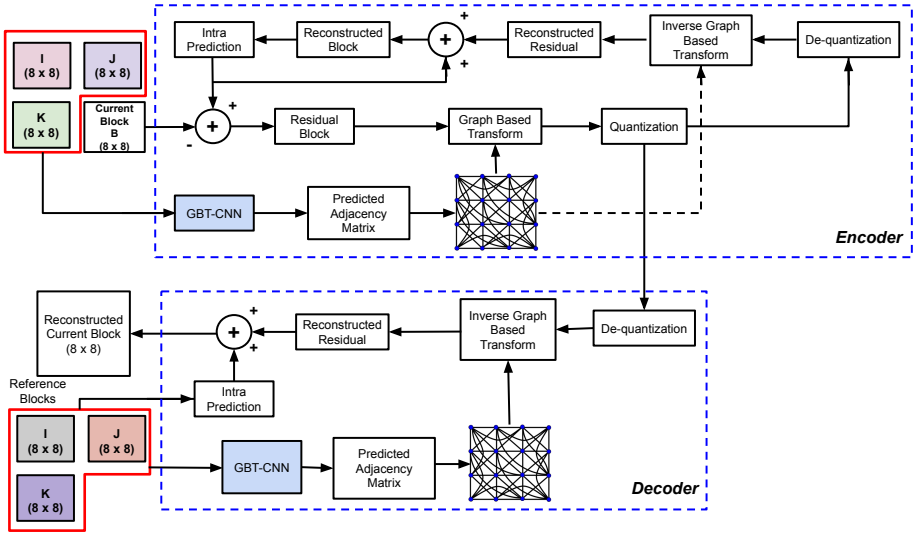

Figure 4: Block diagram of the proposed framework for block-based PTC.

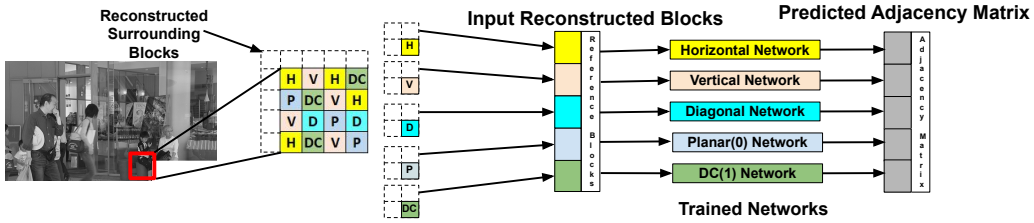

Figure 5: The intra-prediction mode used for each block determines the trained network to use. The figure depicts a section of a frame that has been predicted using several prediction modes.

which are popular video sequences for testing video codecs [3]. We also employ the green (G) component of 10 colour pathology images from the US National Cancer Institute's Center for Biomedical Informatics and Information Technology [13, 14]. We use $64,320$ samples in total for the five networks. Each sample comprises the following values: $\{\mathbf{B}_{[\mathbf{I},\mathbf{J},\mathbf{K}]}, \mathbf{A}_{\mathbf{B}}\}$, where $\mathbf{A}_{\mathbf{B}}$ is the ground truth. 80% of the data is used for training and 20% is used for testing. The training and testing sets do not overlap. We use the Adam optimizer to train each network for 150 epochs with a learning rate of 0.0001 and $\lambda = 0.001$ (see Eq. 4). As mentioned before, the surrounding reconstructed blocks $\mathbf{I}$, $\mathbf{J}$, and $\mathbf{K}$, which are available at the decoder, are used as inputs to a specific trained GBT-CNN according to the mode used by the encoder (see Fig. 5). We use 10, 20, and 40 3D filters respectively in each 3D-CNN layer. For each convolution operation, we apply 1 stride, which leads to feature maps with dimensions of $6 \times 6 \times 10$, $4 \times 4 \times 20$, and $2 \times 2 \times 40$, respectively. At the end of the convolutions, the feature maps are flattened to a vector of dimensions of 160×1. The output layer has 2080 neurons, which matches the upper triangular elements plus those in the diagonal of the matrix $\mathbf{A}_{\mathbf{B}}$.

We compare our proposed method with several GBTs as summarized in Table 1. Specifically, the table tabulates the topology of the graph, the edge weights, and how the graph is obtained to compute the GBT. The KLT, DCT, and DCT/DST as used in the HEVC and VVC standards are also evaluated, with the DCT/DST

being employed as separable transforms for rows and columns of the residual block depending on the prediction mode used. Note that our approach differs from GL-GBT since that method does not use any deep learning. We use the MSE to assess how efficiently the normalized symmetric adjacency matrix is predicted in comparison to the ground truth. Table 2 tabulates the performance of the five trained GBT-CNNs on the test data in terms of the MSE.

We first compute the percentage of PE and the MSE of the reconstructed frames/ images using only a few coefficients under the assumption that no quantization is applied, since the efficiency of a transform is measured by its decorrelating properties and the maximum energy it concentrates in only a few transform coefficients. We set a threshold that indicates the minimum absolute value of the coefficients to be used for the reconstruction. This strategy gradually includes the largest coefficients in a subset by gradually lowering an initial large threshold [10]. Table 3 presents the average PE (%) and MSE values for all evaluated data using a small percentage of the largest coefficients. The GBT-CNN preserves 19.41% and 14.98% more energy than the DCT/DST and the DCT, respectively, if only 5% of the largest coefficients are used. We find that the GBT-L_A outperforms the GBT-CNN; however, since the GBT-L_A requires graph information to compute the inverse transform, this transform is not practical as it significantly increases the overhead. Note that the GL-GBT outperforms all other transforms. Fig. 6 plots the PE (%) and MSE values vs. the percentage of coefficients used for reconstruction for several frames.

We also compute the reconstruction quality attained by the evaluated transforms in terms of the PSNR when quantization is used. Specifically, we employ four quantization parameters (QPs) used by the HEVC and VVC standards: $QP = \{22, 27, 32, 37\}$. Table 4 tabulates average PSNR values for the evaluated frames/ images when these QPs are applied to the transform coefficients. Note that the proposed GBT-CNN outperforms both the DCT/DST and DCT by 7.92 dB and 1.95 dB, respectively, when QP=37. Again, the GBT-L_A outperforms our method by 0.71 db since this method uses actual residuals for reconstruction. Fig. 7 plots the PSNR values for the same frames of Fig. 6.

To demonstrate the rate-distortion trade off among the evaluated transforms, we also compute the transform coding gain in decibels (see Table 4), as the ratio of the distortion incurred between the uncoded and the coded frames [15]:

$$G_T(dB) = 10log_{10}(\frac{D_U}{D_T}), \tag{5}$$

where D_U is the distortion caused by applying direct quantization to the residuals and then dequantizing them to reconstruct the frames, while D_T is the distortion caused by quantization of the transformed coefficients of transform T and then reconstructing the frames after dequantization and inverse transformation. The distrotion is measured in terms of the MSE. Table 4 shows that the GBT-CNN outperforms the DST/DCT and the DCT by 3.78 dB and 3.58 dB respectively, when QP=22. Plots in Fig. 8 show the coding gain of several transforms relative to the KLT, computed as $G_T - G_{KLT}$, where G_T, is the coding gain for transform T and G_{KLT} is the coding gain for KLT. Fig. 9 shows a reconstructed frame of the sequence *BlowingBubble* (Class D) after transformation by several trasnforms and quantization with QP= 37.

Table 1: GBTs used in the evaluation.

Approach	**Explanation**
	All-C Topology
GBT-NN	Train a NN to predict matrix $\mathbf{A_B}$. The graph for GBT has UE weights but no self-loops.
GBT-CNN	**(ours) Train a 3D CNN to predict matrix $\mathbf{A_B}$. The graph for GBT has UE weights but no self-loops.**
GL-GBT	Uses covariance matrices from several training examples to estimate the graph Laplacian.
	4-Connected Topology
GBT-L_A	Use actual residual to compute a graph with UE weights and normalized self-loop weights.
GBT-L_W	Use predicted residuals to predict matrix $\mathbf{A_B}$. The graph for GBT has UE weights with normalized self-loop weigths.

Table 2: Performance evaluation of the model on test data for all the networks.

	Networks				
Metric	H	V	D	DC	P
MSE	0.0076	0.0162	0.0134	0.0185	0.0384

As depicted, the GBT-CNN achieves a higher visual reconstruction quality than the DCT. The GL-GBT achieves a visual quality very close to that achieved by the KLT.

Computational complexity: Any GBT requires eigendecomposition of the Laplacian graph. However, the eigendecomposition used by the KLT tends to be more complex as it uses a dense matrix. On the other hand, the sparsity in the graph Laplacian for the GBT can be controlled by the graph topology, which can lead to a lower computational complexity. Unfortunately, the GBT is just as computationally expensive as the KLT for the case of the All-C topology. In terms of learnable parameters, the network used by the GBT-CNN requires 4723510 parameters. The architecture used by the GBT-NN [11] requires 22368256 learnable parameters.

4. Conclusion

In this paper, we proposed the GBT-CNN, a new class of GBTs that performs efficiently in block-based PTC with intra-prediction. The GBT-CNN is based on a 3D-CNN that learns a mapping function to approximate a symmetric adjacency matrix associated with the graph of the residual block to be encoded. We evaluated the performance of the GBT-CNN 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.

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

	Percentage of coefficients used					
	1%		5%		10%	
	PE	MSE	PE	MSE	PE	MSE
GL-GBT [8]	53.23	45.18	**92.37**	**07.66**	**95.92**	**06.81**
KLT	**55.51**	**44.49**	90.85	10.22	93.56	07.97
DCT	16.84	82.99	52.11	48.38	69.07	32.03
DCT/DST	16.14	83.69	50.18	50.64	66.88	34.24
GBT-NN [11]	18.97	78.72	55.43	44.46	72.40	28.94
GBT-CNN (ours)	**21.45**	**76.36**	**59.92**	**39.12**	**73.16**	**27.9**
GBT-L_A [10]	24.71	75.17	60.47	40.21	74.28	26.47
GBT-L_W [10]	17.01	82.82	52.58	47.93	69.18	31.86

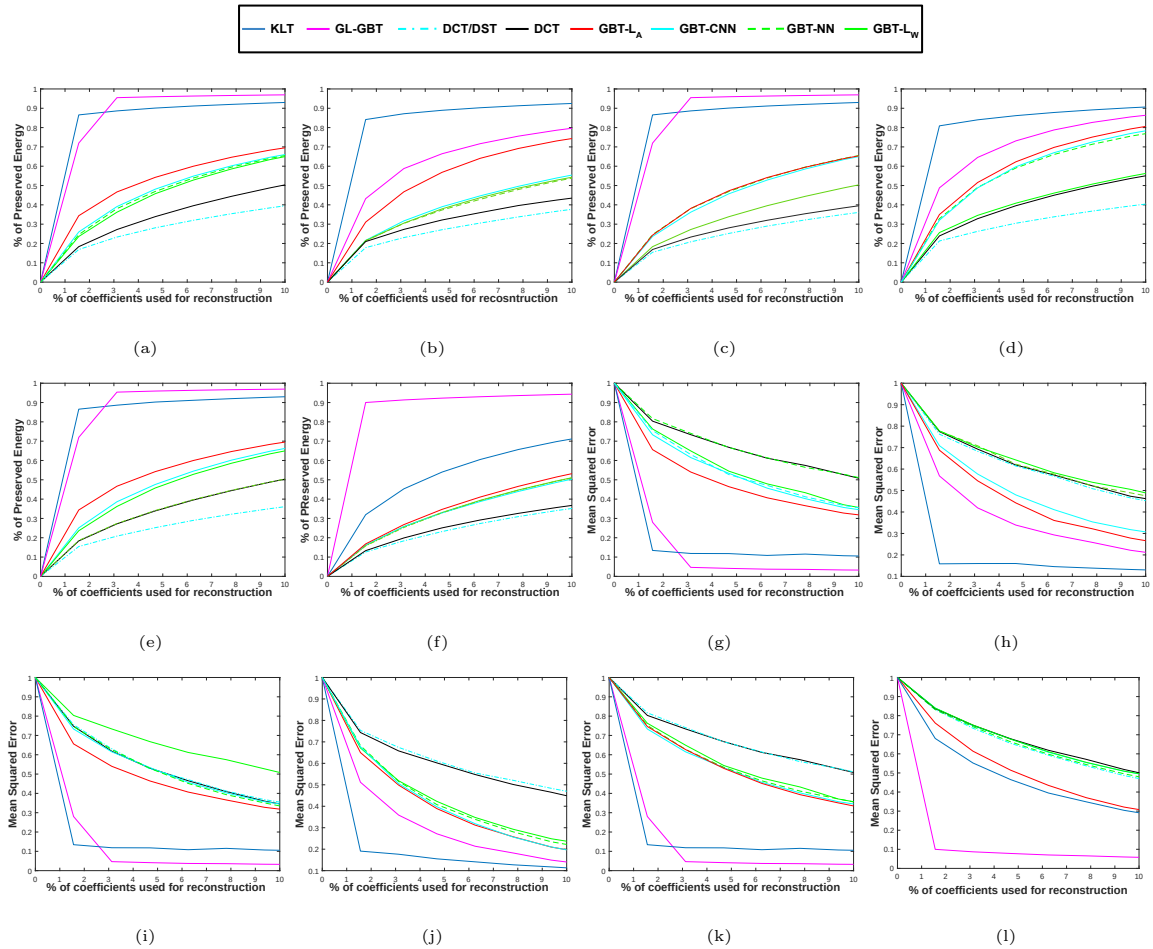

Figure 6: PE (%) and MSE vs. percentage of coefficient used for reconstruction of a frame of sequence {(a), (g)} *BQTerrace*(Class B), {(b), (h)}*BQMall*(Class C), {(c), (i)}*ChinaSpeed*(Class SC), {(d), (j)}*RaceHorse*(Class D), {(e), (k)}*PeopleOnStreet*(Class A), {(f), (l)}*KristineAndSara*(Class E).

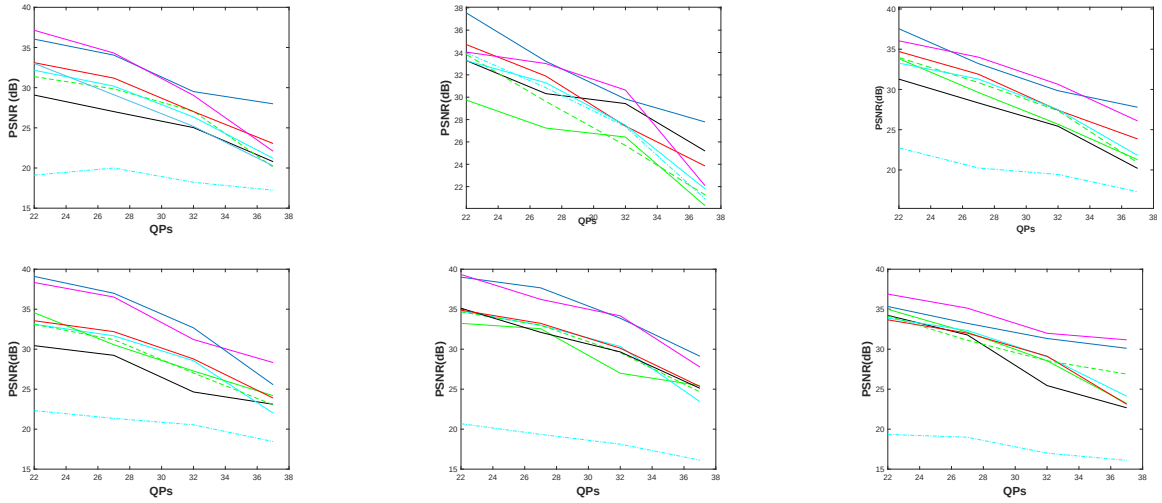

Figure 7: PSNR (dB) vs. QP for a frame of sequence (left to right and top to bottom) *BQTerrace* (Class B), *BQMall* (Class C), *ChinaSpeed* (Class SC), *RaceHorse* (Class D), *PeopleOnStreet* (Class A), *KristineAndSara* (Class E). The legend for this figure is the same as the one for Fig. 6.

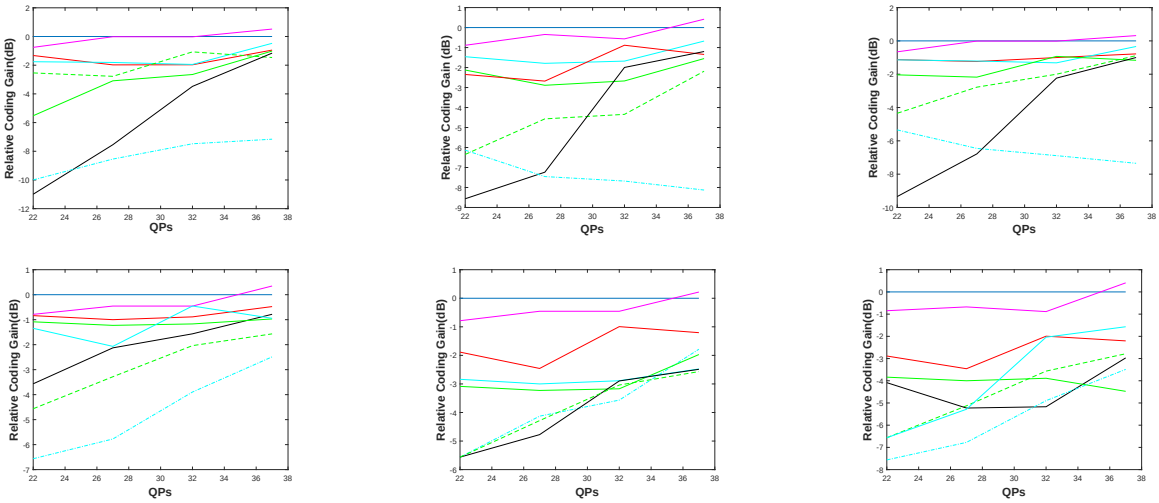

Figure 8: Relative coding gain vs. QP for a frame of sequence (left to right and top to bottom) *BQTerrace* (Class B), *BQMall* (Class C), *ChinaSpeed* (Class SC), *RaceHorse* (Class D), *PeopleOnStreet* (Class A), *KristineAndSara* (Class E). The legend for this figure is the same as the one for Fig. 6.

Table 4: Average PSNR and coding gain when using quantization on the transform coefficients.

	Quantization Parameters							
	QP=22		QP=27		QP=32		QP=37	
	PSNR	Gain	PSNR	Gain	PSNR	Gain	PSNR	Gain
GL-GBT	39.63	5.80	**36.92**	**7.68**	**33.05**	**8.66**	28.45	7.50
KLT	**40.22**	**6.58**	36.05	7.66	32.71	8.43	**29.62**	**8.03**
DCT	35.21	-1.63	31.02	0.77	28.29	0.95	23.07	0.99
DCT/DST	20.56	-1.83	19.02	0.45	18.28	0.13	17.10	1.48
GBT-NN	35.86	1.03	31.69	2.43	29.18	2.80	23.93	3.22
GBT-CNN (ours)	**36.13**	**1.95**	**32.73**	**2.65**	**29.90**	**2.91**	**25.02**	**3.25**
GBT-L_A	36.69	2.46	33.84	4.32	30.31	6.42	25.73	7.58
GBT-L_W	35.71	-0.46	31.58	-0.72	28.55	0.16	23.16	1.54

We also compared the coding gain of the evaluated transforms. The evaluation results show that the proposed GBT-CNN outperforms the DCT and the DCT/DST, while the GL-GBT achieves the best performance, surpassing the KLT.

References

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and 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*, vol. 30, no. 3, pp. 83–98, 2013.

[2] J. Lainema, F. Bossen, W.J. Han, J. Min, and K. Ugur, "Intra coding of the hevc standard," *IEEE Trans. Circuits and Systems for Video Tech*, vol. 22, no. 12, pp. 1792–1801, 2012.

[3] W. Hu, G. Cheung, and A. Ortega, "Intra-prediction and generalized graph fourier transform for image coding," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1913–1917, 2015.

[4] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[5] D. Roy, T. Guha, and V. Sanchez, "Graph-based transform with weighted self-loops for

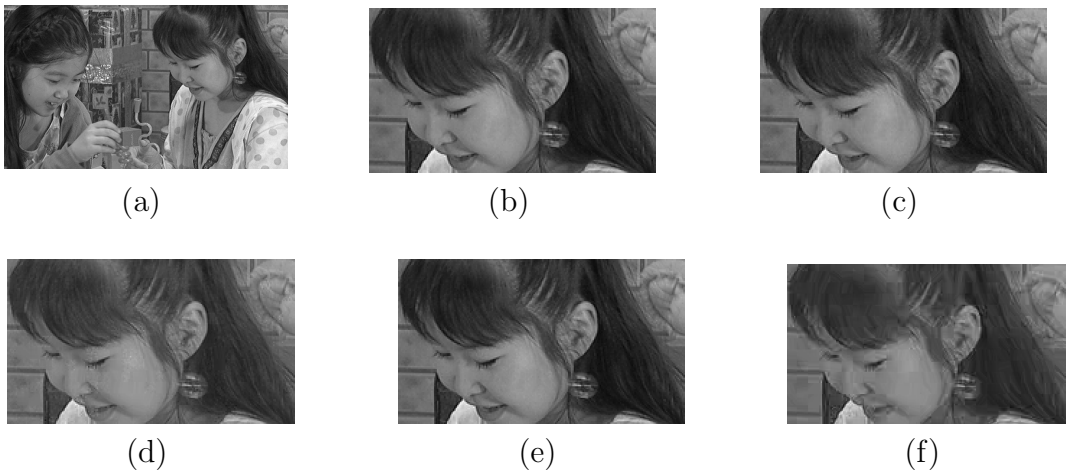

Figure 9: (a) An original frame of sequence *BlowingBubble* (Class D). (b) An area reconstructed after using the KLT (PSNR = 28.32 dB), (c) the proposed GBT-CNN (PSNR = 24.95 dB), (d) the GBT-NN (PSNR = 23.57 dB), (e) the GL-GBT (PSNR = 28.13 dB), and (f) the DCT (PSNR = 22.38). In all cases, QP=37.

predictive transform coding based on template matching," in *2019 Data Compression Conference (DCC)*, 2019, pp. 329–338.

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

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

[8] H. E. Egilmez, Y. H. Chao, and A. Ortega, "Graph-based transforms for video coding," *IEEE Transactions on Image Processing*, vol. 29, pp. 9330–9344, 2020.

[9] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825–841, 2017.

[10] D. Roy, T. Guha, and V. Sanchez, "Graph based transforms based on graph neural networks for predictive transform coding," in *2021 Data Compression Conference (DCC)*, 2021, pp. 367–367.

[11] D. Roy, T. Guha, and 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.

[12] S. Boyd, "Convex optimization of graph laplacian eigenvalues," *In Proc.Int. Congr. Math., volume 3*, vol. 3, no. 1, pp. 1311–1319, 2006.

[13] K. Tomczak, P. Czerwińska, and M. Wiznerowicz, "The cancer genome atlas (tcga): an immeasurable source of knowledge," *Contemporary oncology*, vol. 19, no. 1A, pp. A68, 2015.

[14] V. Sanchez, F. Aulí-Llinàs, J. Bartrina-Rapesta, and 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.

[15] J. Han, A. Saxena, V. Melkote, and K. Rose, "Jointly optimized spatial prediction and block transform for video and image coding," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1874–1884, 2012.