EE6427 24S1

1. In a new image compression scheme, a student would like to follow similar steps in the baseline JPEG to perform grayscale image compression. The student proposes to partition an image into multiple 4×4 pixel blocks, perform 4×4 Discrete Cosine Transform (DCT) for each pixel block, followed by corresponding quantization and entropy encoding.

The two-dimensional DCT (2-D DCT) matrix of a 4×4 pixel block is given by

$$T = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix},$$

where $a = \frac{1}{2}$, $b = \frac{1}{\sqrt{2}}\cos\frac{\pi}{8}$, and $c = \frac{1}{\sqrt{2}}\cos\frac{3\pi}{8}$.

(a) Find the 2-D DCT of the following pixel block **A**. Round your answer to 2 decimal places.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 50 & 50 & 0 \\ 0 & 50 & 50 & 0 \end{bmatrix}$$

1.(a) $\left(\cos\frac{\pi}{8} = \frac{\sqrt{2+\sqrt{2}}}{2} \quad \cos\frac{3\pi}{8} = \frac{\sqrt{2-\sqrt{2}}}{2}\right)$

F = 2D-DCT of A

$= T \cdot A \cdot T^T$

$\left(b+c = \cos\frac{\pi}{8}, \; b-c = \cos\frac{3\pi}{8}\right)$

$$= \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 50 & 50 & 0 \\ 0 & 50 & 50 & 0 \end{bmatrix} \cdot T^T$$

$$= \begin{bmatrix} 0 & 50 & 50 & 0 \\ 0 & -50(b+c) & -50(b+c) & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 50(b-c) & 50(b-c) & 0 \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

$$= \begin{bmatrix} 50 & 0 & -50 & 0 \\ -50(b+c) & 0 & 50(b+c) & 0 \\ 0 & 0 & 0 & 0 \\ 50(b-c) & 0 & -50(b-c) & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 50.00 & 0 & -50.00 & 0 \\ -46.19 & 0 & 46.19 & 0 \\ 0 & 0 & 0 & 0 \\ 19.13 & 0 & -19.13 & 0 \end{bmatrix}$$

(b) The 2-D DCT of pixel block **A** goes through quantization defined by the quantization matrix **Q** below:

$$\mathbf{Q} = \begin{bmatrix} 15 & 30 & 40 & 50 \\ 30 & 40 & 50 & 60 \\ 40 & 50 & 60 & 70 \\ 50 & 60 & 70 & 80 \end{bmatrix}.$$

Find the quantized DCT coefficients of pixel block **A**.

(b) $\hat{F} = \text{round}\left(\frac{F(i,j)}{Q(i,j)}\right)$

$$= \begin{bmatrix} 3 & 0 & -1 & 0 \\ -2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

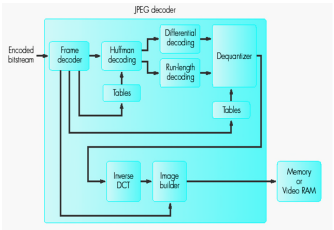
(c) A student claims that: "*Karhunen Loeve Transform (KLT) can achieve better compression ratio than DCT in transform-based coding. Hence KLT is a more practical and faster method to compress a wide range of different images.*" State clearly whether you **agree** or **disagree** with the claim, and briefly justify your answer.

I disagree.
Although the KLT can theoretically achieve better compression, it is impractical and slow since its basis functions depend on the image statistics and requires eigenvalue computation.
In contrast the DCT uses fixed and not image-
redund
Therefo thod.

Why DCT?

- 2D-DCT is a popular transform used in transform-based image compression.
- It can offer the following:
  - Energy compaction for transform coefficients
  - Redundancy reduction amongst transform coefficients
- Pro: good compression results, basis functions are fixed and not image-dependent.
- Con: compression is not as effective as some other transforms, e.g., Karhunen Loeve Transform.

JPEG Decoder



(d) The 2-D

where p, q, r and s are real numbers.

A user would like to obtain a normalized pixel block $\mathbf{D}_N$ by subtracting the average intensity of pixel block **D** from each pixel in the pixel block **D**. Assuming that the average intensity of pixel block **D** is m, where m is a real number, find the 2-D DCT of the normalized pixel block $\mathbf{D}_N$.

$D_N = D - m J_{4\times4}$

$\therefore$ 2D-DCT of D_N

= 2D-DCT of D

$$- \begin{bmatrix} 4m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad p = 4m$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Consider a simplified H.264 4×4 intra coding that uses only Mode 0 (Vertical), Mode 1 (Horizontal), and Mode 2 (DC).

A 4×4 luma pixel block shown by the indicated bounding box, together with some neighboring pixels, is shown in Figure 1 below.

10	20	20	40	40
20	20	20	40	40
20	20	20	40	40
20	20	20	30	30
20	20	20	30	30

Figure 1

(a) Find the prediction errors for Mode 0, Mode 1, and Mode 2, respectively.

2.(a)

Mode 0 predictions $\begin{bmatrix} 20 & 20 & 40 & 40 \\ 20 & 20 & 40 & 40 \\ 20 & 20 & 40 & 40 \\ 20 & 20 & 40 & 40 \end{bmatrix}$ Errors $\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -10 & -10 \\ 0 & 0 & -10 & -10 \end{bmatrix}$

Mode 1 $\begin{bmatrix} 20 & 20 & 20 & 20 \\ 20 & 20 & 20 & 20 \\ 20 & 20 & 20 & 20 \\ 20 & 20 & 20 & 20 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 20 & 20 \\ 0 & 0 & 20 & 20 \\ 0 & 0 & 10 & 10 \\ 0 & 0 & 10 & 10 \end{bmatrix}$

Mode 2 $\begin{bmatrix} 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 \\ 25 & 25 & 25 & 25 \end{bmatrix}$ $\begin{bmatrix} -5 & -5 & 20 & 20 \\ -5 & -5 & 20 & 20 \\ -5 & -5 & 10 & 10 \\ -5 & -5 & 10 & 10 \end{bmatrix}$

(b) Find the sum of absolute errors for each prediction error using Mode 0, Mode 1, and Mode 2 in part (a). Hence write down the smallest prediction error $E(i, j)$ for the indicated 4×4 pixel block based on the metric of sum of absolute errors.

2.(b) Mode 0: MAE = 40

Mode 1: MAE = 20×4 + 10×4 = 120

Mode 2: MAE = 5×8 + 20×4 + 10×4 = 160

∴ 40 < 120 < 160

∴ smallest prediction error $E(i,j) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 10 \\ 0 & 0 & -10 & -10 \end{bmatrix}$ (Mode 0)

(c) The smallest prediction error $E(i, j)$ undergoes the Integer Transform with transform matrix given by:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}.$$

Find the Integer Transform of the smallest prediction error $E(i, j)$.

(7 Marks)

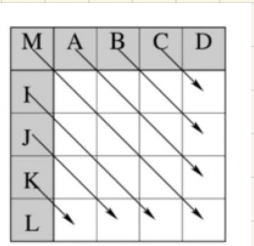
2.(c) $D(i,j) = H \cdot E(i,j) \cdot H^T$

$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 10 \\ 0 & 0 & -10 & -10 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -40 & 60 & 0 & -20 \\ 60 & -90 & 0 & 30 \\ 0 & 0 & 0 & 0 \\ -20 & 30 & 0 & -10 \end{bmatrix}$

(d) Draw a simple diagram that shows Mode 4 (Diagonal Down-Right) used in the intra coding of H.264.

(6 Marks)

2.(d)



3. (a) A Vanilla Recurrent Neural Network (RNN) has the following settings.

Initial hidden state, $\mathbf{h}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$,

Hidden state weight matrix, $\mathbf{W}_{hh} = \begin{bmatrix} 0.5 & 0.2 \\ 0.3 & 0.1 \end{bmatrix}$,

Input weight matrix, $\mathbf{W}_{xh} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}$,

Output weight matrix, $\mathbf{W}_{hy} = [0.3 \quad 0.2]$.

Assume no bias is used in the computation of the RNN.

A 2-timestep input is given by $\mathbf{x} = [\mathbf{x}_1 \quad \mathbf{x}_2]$ where $\mathbf{x}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\mathbf{x}_2 = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$.

The ground truth output sequence is given by $\mathbf{g} = [g_1 \quad g_2]$ where $g_1 = 0.5$ and $g_2 = 0.8$.

(i) Find the hidden state $\mathbf{h}_1$ at timestep $t = 1$.

3.(a).(i) $h_t = \tanh(W_{hh} \cdot h_{t-1} + W_{xh} \cdot X_t)$

$\therefore h_1 = \tanh(W_{hh} \cdot h_0 + W_{xh} X_1)$

$= \tanh\left(\begin{bmatrix} 0.5 & 0.2 \\ 0.3 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right)$

$= \tanh\left(\begin{bmatrix} 0.4 \\ 1 \end{bmatrix}\right) \approx \begin{bmatrix} 0.3799 \\ 0.7616 \end{bmatrix}$

(ii) Find the hidden state $\mathbf{h}_2$ at timestep $t = 2$.

3.(a).(ii) $h_2 = \tanh(W_{hh} \cdot h_1 + W_{xh} \cdot X_2)$

$= \tanh\left(\begin{bmatrix} 0.5 & 0.2 \\ 0.3 & 0.1 \end{bmatrix} \cdot \begin{bmatrix} 0.3799 \\ 0.7616 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}\right)$

$= \tanh\left(\begin{bmatrix} 1.2422 \\ 2.4901 \end{bmatrix}\right) \approx \begin{bmatrix} 0.8461 \\ 0.9863 \end{bmatrix}$

(iii) Find the mean squared error between the calculated output sequence and the ground truth output sequence.

3.(a).(iii) $y_t = W_{hy} \cdot h_t$

$\therefore y_1 = W_{hy} \cdot h_1 = [0.3 \ 0.2] \cdot \begin{bmatrix} 0.3799 \\ 0.7616 \end{bmatrix} = 0.26629$

$y_2 = W_{hy} \cdot h_2 = [0.3 \ 0.2] \cdot \begin{bmatrix} 0.8461 \\ 0.9863 \end{bmatrix} = 0.45109$

$\therefore MSE = \frac{1}{2}[(g_1 - y_1)^2 + (g_2 - y_2)^2]$

$= \frac{1}{2} \times [(0.5 - 0.26629)^2 + (0.8 - 0.45109)^2]$

≈ 0.0882

(b) A user would like to develop a video action recognition application using a vanilla RNN model. He proposes to use a Convolutional Neural Network (CNN) to extract feature embedding for each frame of the video, and then use a vanilla RNN to perform the video action recognition. Each video clip for this application has a duration of 10 seconds, with a frame rate of 25 frames per second.

Suggest the change(s) that should be made to the RNN structure in part (a) to perform the video action recognition.

3.(b) total number of frames = 10 s × 25 frame/s = 250 frames

total number of feature vectors

① Input Modification: $(X_{1\times 2} \Rightarrow X_{1\times 250})$

250 sequential feature vectors instead of pixel values will be fed into the RNN as the sequence input

② Output Modification: (y / y_{250})

The RNN can output a single class probability vector, which is the output from the last timestep) for video-level action recognition or a sequence of class probabilities for each frame for frame-level action recognition.

③ Hidden Stage Change: $(h_0, h_1, \ldots h_{250})$

④ Fully Connected Layer:

A fully connected with a softmax activation function is added for classification