視訊處理作業報告

HW1
Huffman coding and Huffman Coding with Pixel Prediction

授課教授：柳金章
學　　生：楊憲閔
學　　號：613410047
Due date：2024/04/30
Date hand in：2024/04/25

目錄

Technical description

1. Huffman coding

Huffman Coding 是一種無失真的資料壓縮技術，廣泛應用於影像與文字檔案中。其核心概念在於根據每個符號出現的機率分配不等長的二進位碼：機率高者分配較短的編碼，機率低者則分配較長的編碼，藉此達到整體資料長度的縮減。實作流程如下：

(1) 統計符號機率：分析輸入資料中所有像素值的出現頻率。

(2) 建構 Huffman Tree：依據機率建立一棵二元樹。從最低機率的兩個節點合併為新節點，並將其總機率重新加入佇列。反覆執行直到只剩一個根節點。

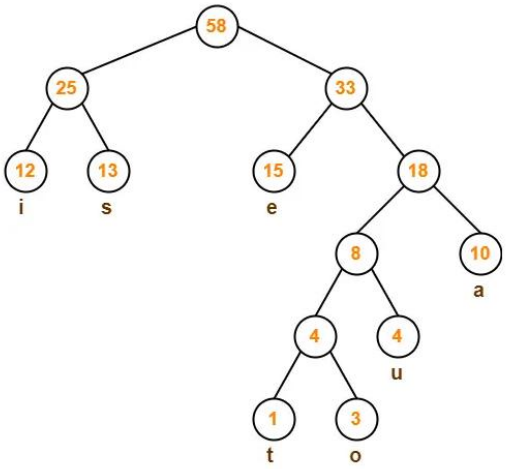
(3) 產生 Huffman 編碼表：從根節點開始，左子節點編碼為 0，右子節點為 1，依此遞迴直至樹葉節點，得到每個像素對應的二進位碼。

(4) 資料編碼與儲存：將原始資料轉換為 Huffman 編碼後的二進位流，同時儲存編碼表以便日後解碼。

(5) 解碼流程：讀取編碼表與壓縮後的二進位資料，逐位對照還原成原始像素資料。

此方法可有效壓縮資訊量大、符號分布不均的資料，並確保在

解壓時能無損還原。



圖(1) Huffman Tree。

Characters	Frequencies
a	10
e	15
i	12
o	3
u	4
s	13
t	1

圖(2) 符號與頻率表。

2. Huffman coding with pixel prediction

為進一步提高壓縮率，可搭配像素預測技術使用 Huffman 編碼。此方法廣泛用於影像壓縮中，其原理在於預測當前像素的值，再計算與實際像素的差值（prediction residual），用來進行 Huffman 編碼。由於差值分布集中，entropy 較低，因此可得到更短

的總編碼長度。步驟如下：

(1) 進行像素預測：對每個像素，根據鄰近像素（如上方、左方、左上角）進行估算。常見預測模式如圖中所示。

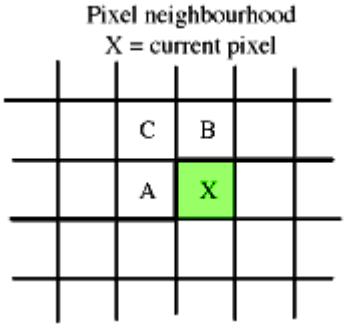
(2) 計算差值：將預測值與真實值相減，得到 residual 資料。

(3) 統計機率與編碼：計算 residual 的出現頻率，建立 Huffman Tree 並產生對應編碼表。

(4) 壓縮與儲存：以 Huffman 編碼取代差值資料，同時保存預測模式、參數與編碼表。

(5) 解壓與還原：解碼 residual 後，加回預測值以重建原始影像。

此方法在多數影像中具有更佳的壓縮效果，尤其是圖像中具強連續性或平滑區域時。雖然此技術可能因預測誤差導致微小失真，但通常在人眼感知上無明顯差異。



Encoder type	Prediction method
0	no prediction
1	A
2	B
3	C
4	A + B – C
5	A + ((B – C) / 2)
6	B + ((A – C) / 2)
7	(A + B) / 2

圖(3) 左側為預測時所用的相鄰 pixel，X 為目前的 pixel，A、B、C 為相鄰的 pixel。右側為預測方法。

Experimental results

1. 程式執行流程：

(1) 確保已安裝相關 module，本次作業使用 module 如下所示:

```
# Required Libraries
import numpy as np
from PIL import Image
from collections import Counter, defaultdict
import heapq
import os , cv2
import matplotlib.pyplot as plt
```

圖(4) 會使用到的 module

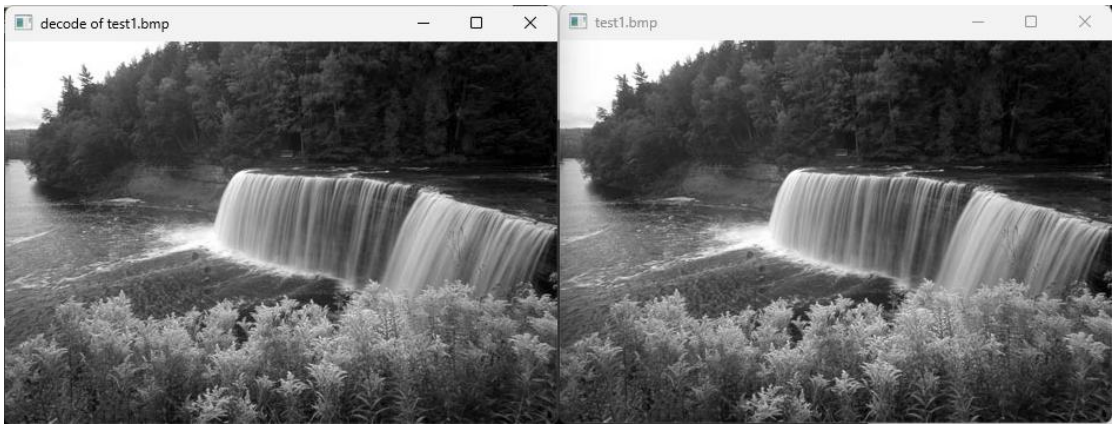
(2) 進到作業的目錄底下，會看到兩張 test images、兩個.py 檔及此 pdf 檔。HW1_1.py 對應於 Huffman coding。HW1_2.py 對應於 Huffman coding with pixel prediction。點右鍵按在終端中開啟，輸入 python HW1_1.py(或是 HW1_2.py)，程式即開始執行。

(3) 程式會讀取兩張圖片並對其做 Huffman coding。HW1_1.py 會印出第一張圖片的 code book、壓縮率，並顯示原圖片與解壓縮後還原的影像，關閉視窗後會進行第二張的，過程與第一張類似。HW1_2.py 與 HW1_1.py 差別在印出與輸出的數量不同，第二

題為 8 個，一一對應到不同的 prediction mode。關閉視窗後會進行第二張的，過程與第一張類似。

2. 程式執行結果：

(1)Huffman coding



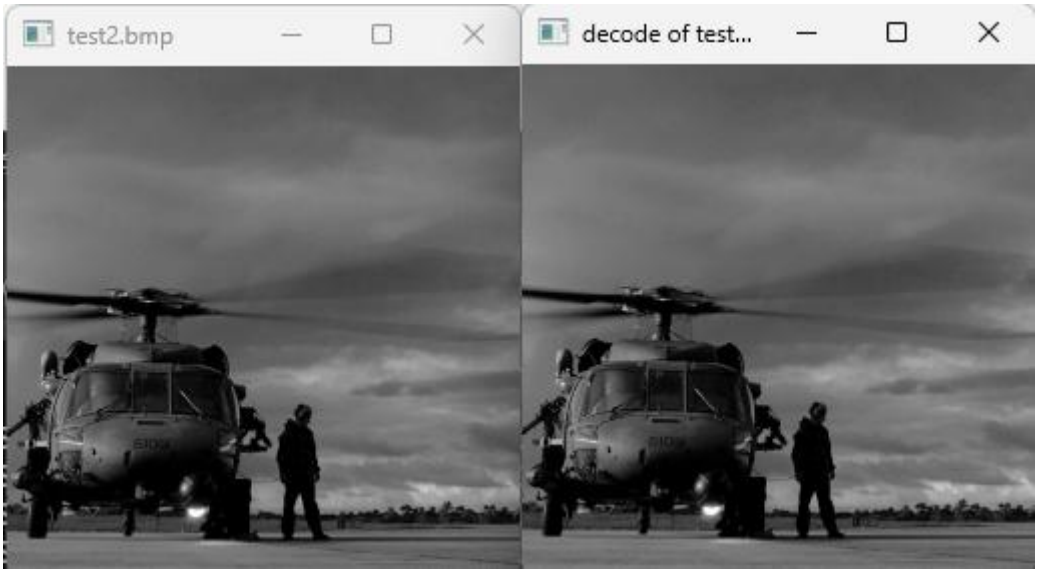
圖(5) test1.bmp 原圖與 decode 後的影像

```
test1.bmp
Codebook:
Symbol: 0, Freq: 0.0012176759410801965, Code: 1110000010
Symbol: 1, Freq: 0.0005106382978723404, Code: 11000011000
Symbol: 2, Freq: 0.0029590834697217675, Code: 00001010
Symbol: 3, Freq: 0.0021865793780687396, Code: 110011110
Symbol: 4, Freq: 0.0021669394435351883, Code: 110011010
Symbol: 5, Freq: 0.002736497545008183, Code: 111110000
Symbol: 6, Freq: 0.0022389525368248773, Code: 110100101
Symbol: 7, Freq: 0.0032471358428805236, Code: 00101001
Symbol: 8, Freq: 0.002579378068739771, Code: 111001001
Symbol: 9, Freq: 0.0031947626841243863, Code: 00100110
Symbol: 10, Freq: 0.0033453355155482816, Code: 00110011
Symbol: 11, Freq: 0.0028477905073649753, Code: 111111110
Symbol: 12, Freq: 0.00332569558101473, Code: 00110010
Symbol: 13, Freq: 0.0035286641571194763, Code: 01000011
Symbol: 14, Freq: 0.0026186579378068746, Code: 111011011
Symbol: 15, Freq: 0.0060621931260229133, Code: 0001000
Symbol: 16, Freq: 0.0074631751227495913, Code: 0111011
Symbol: 17, Freq: 0.0072864157119476274, Code: 0101111
Symbol: 18, Freq: 0.0069656301145662854, Code: 0100000
Symbol: 19, Freq: 0.0071751227495908354, Code: 0101000
Symbol: 20, Freq: 0.0068936170212765954, Code: 0011111
Symbol: 21, Freq: 0.0068085106382978724, Code: 0011100
Symbol: 22, Freq: 0.0062324058919803599, Code: 0001011
```

圖(6) test1.bmp 印出的 codebook

```
Symbol: 252, Freq: 0.0012438625204582651, Code: 1110000011
Symbol: 253, Freq: 0.002520458265139116, Code: 111000100
Symbol: 254, Freq: 0.004739770867430442, Code: 11011011
Symbol: 255, Freq: 0.02092962356792144, Code: 111010
Compression Ratio: 1.0473
------------------------------------------------------------
```

圖(7) test1.bmp 印出的 compression ratio



圖(8) test1.bmp 原圖與 decode 後的影像

```
test2.bmp
Codebook:
Symbol: 0, Freq: 0.0214691162109375, Code: 111011
Symbol: 1, Freq: 0.01763916015625, Code: 101111
Symbol: 2, Freq: 0.009246826171875, Code: 1100010
Symbol: 3, Freq: 0.006072998046875, Code: 0001110
Symbol: 6, Freq: 0.00396728515625, Code: 10011011
Symbol: 7, Freq: 0.0028076171875, Code: 111100111
Symbol: 8, Freq: 0.0038604736328125, Code: 10011010
Symbol: 9, Freq: 0.0040130615234375, Code: 10101011
Symbol: 10, Freq: 0.0031280517578125, Code: 00110010
Symbol: 11, Freq: 0.003326416015625, Code: 01001101
Symbol: 12, Freq: 0.0029144287109375, Code: 111111000
Symbol: 14, Freq: 0.0030517578125, Code: 00011110
Symbol: 15, Freq: 0.0032806396484375, Code: 01001100
Symbol: 17, Freq: 0.0034942626953125, Code: 01101001
Symbol: 18, Freq: 0.002777099609375, Code: 111100110
Symbol: 19, Freq: 0.003143310546875, Code: 01000000
Symbol: 20, Freq: 0.00286865234375, Code: 111101100
Symbol: 21, Freq: 0.0023956298828125, Code: 110011011
Symbol: 22, Freq: 0.002227783203125, Code: 101110011
Symbol: 23, Freq: 0.0023040771484375, Code: 110001100
Symbol: 24, Freq: 0.0018463134765625, Code: 100001011
Symbol: 26, Freq: 0.001922607421875, Code: 100100001
Symbol: 28, Freq: 0.00225830078125, Code: 110000011
Symbol: 29, Freq: 0.0020751953125, Code: 101110000
```

圖(9) test2.bmp 印出的 codebook

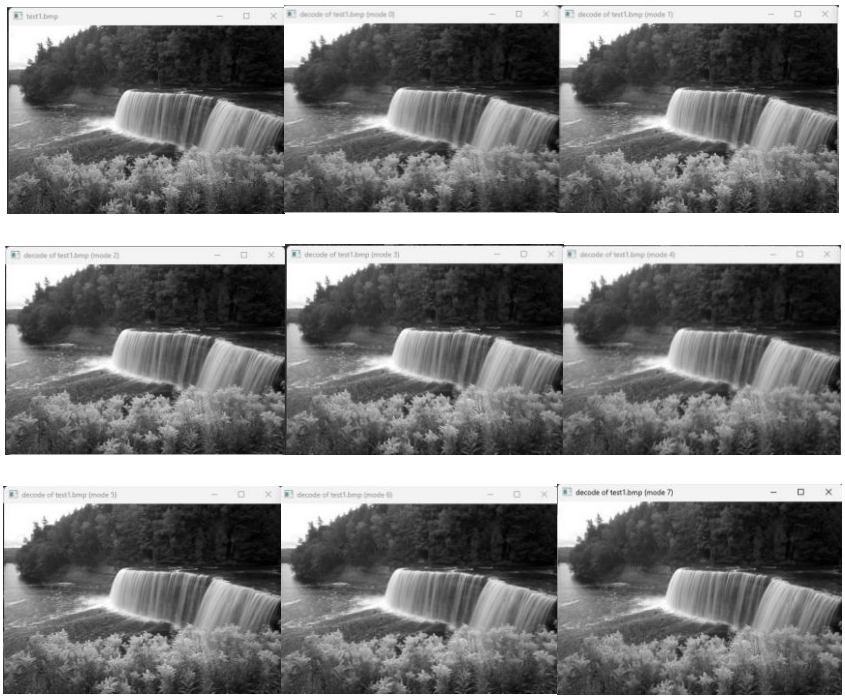
```
Symbol: 245, Freq: 3.0517578125e-05, Code: 101110010000011
Symbol: 248, Freq: 3.0517578125e-05, Code: 101110010000010
Symbol: 251, Freq: 6.103515625e-05, Code: 10111001000000
Symbol: 255, Freq: 7.62939453125e-05, Code: 10111001000101
Compression Ratio: 1.2112
------------------------------------------------------------
```

圖(10) test2.bmp 印出的 compression ratio

從圖(5)和圖(8)中可以看到原圖與解碼後影像看起來一樣。

(2) Huffman coding with pixel prediction

（因 codebook 太多了且與第一題類似就不一一截圖）



圖(11) 對 test1.bmp 以不同的預測方式做 Huffman coding with pixel prediction 的結果。左上為原始影像，其餘為不同 mode 解壓縮後的影像。

```
Compression Ratio (mode: 0): 1.0473
------------------------------------------------------------
```

```
Compression Ratio (mode: 1): 2.6097
------------------------------------------------------------
```

```
Compression Ratio (mode: 2): 2.5491
```

```
Compression Ratio (mode: 3): 2.4115
```

```
Compression Ratio (mode: 4): 1.7248
```

```
Compression Ratio (mode: 5): 3.0315
```

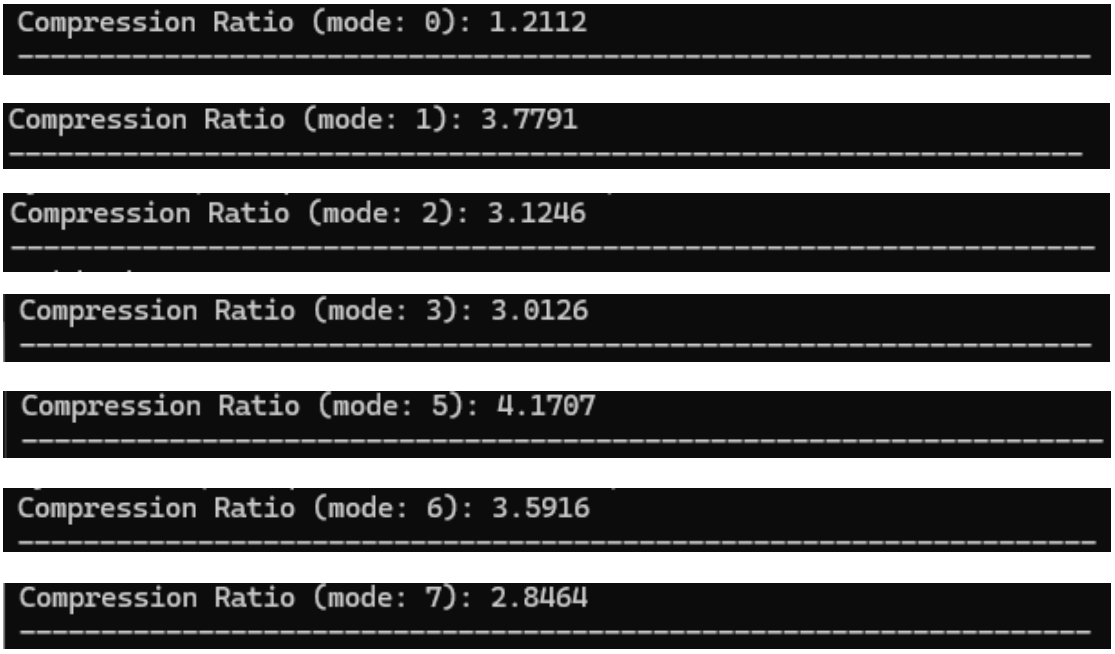
```
Compression Ratio (mode: 6): 2.9852
```

```
Compression Ratio (mode: 7): 1.8851
```

圖(12) 對 test1.bmp 以不同的預測方式做 Huffman coding with pixel prediction 後的壓縮比。

圖(13) 對 test2.bmp 以不同的預測方式做 Huffman coding with pixel prediction 的結果。左上為原始影像，左上為原

始影像，其餘為不同 mode 解壓縮後的影像。



圖(14) 對 test2.bmp 以不同的預測方式做 Huffman coding with pixel prediction 後的壓縮比。

從圖(11)和圖(13)可以看出經過 Huffman coding with pixel prediction 進行壓縮並解壓後，影像並沒有消失，雖然在一些 pixel 的計算上可能有些誤差，但從肉眼觀察上，和原始影像並沒有相差太大的地方，大致上有還原回來。

Discussions

本次作業比較困難的點是需要時刻注意程式裡面參數的資料型態，避免一不小心就輸入到錯誤的資料型態，導致程式無法執行，此外在第二題中 predict 的 gray level 值也須注意，不可超過 0~255，不然就不合理並導致生成錯誤的 codebook。

從輸出的.bin 檔可以得知，一般的 Huffman coding 雖然可以將影像壓縮，但並沒有壓縮很多，好處是解壓縮時影像有辦法完全還原。而 Huffman coding with pixel prediction 的壓縮效果就比較明顯了，因為其運用相鄰 pixel 做預測以獲得差值的關係，固有許多差值其實是相同的，其所構建出 Huffman tree 的深度就比較淺，故 Huffman code 的二進制編碼比較短，壓縮效果比較好，但缺點就是因其有許多差值相同的關係，故解壓縮時影像會有些許的誤差，不過大致上都屬於在肉眼看不出差異的範圍內。

References and Appendix

https://www.gatevidyalay.com/huffman-coding-huffman-encoding/

https://www.geeksforgeeks.org/huffman-coding-in-python/