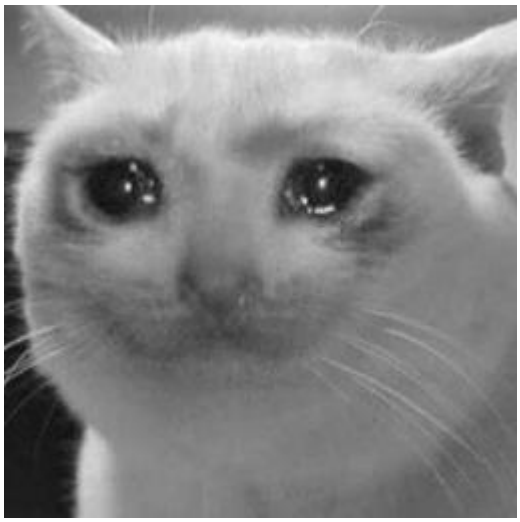
學號:B06902136
系級:資工四
姓名:賴冠毓

Problem 1: DIGITAL HALFTONING
(a)
My motivation and approach:
先依據講義的公式算出 threshold matrix，再將 image 適當擴展，跟 threshold matrix 進行比較，最後將比較完的結果壓回原本圖片的大小。

- **Dithering**
  - **Determine the threshold matrix**

$$T(i,j) = 255 \cdot \frac{I(i,j) + 0.5}{N^2}$$

Original images:



Output images:

Discussion of results:

確實有些微達到 dither 的效果，但顆粒不明顯。

(b)

My motivation and approach:

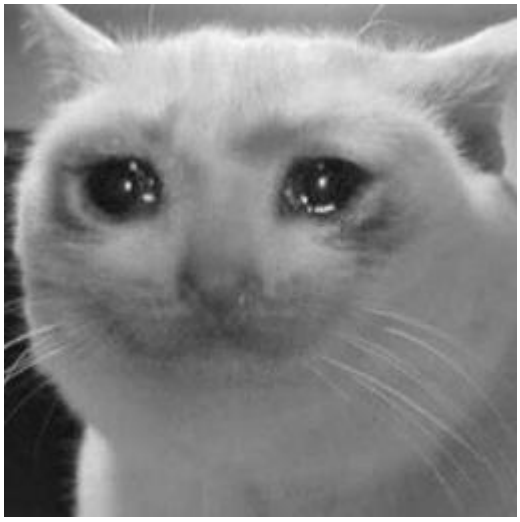
做法跟前一小題一樣，只是要先把 dither matrix 從 2 * 2 擴大為 256 * 256

擴大的方法如講義公式。

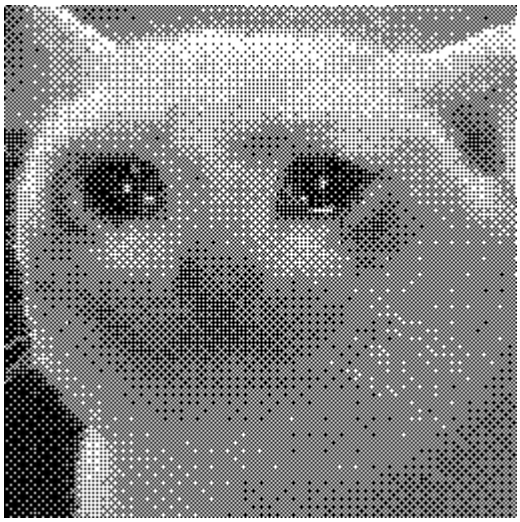
- **The general form of the NxN dither matrix**
  - **2x2 → 4x4 → 8x8 → 16x16…**

$$I_{2n}(i,j)=\begin{bmatrix} 4I_n(i,j)+1 & 4I_n(i,j)+2 \\ 4I_n(i,j)+3 & 4I_n(i,j)+0 \end{bmatrix}$$

Original images:



Output images:



Discussion of results:

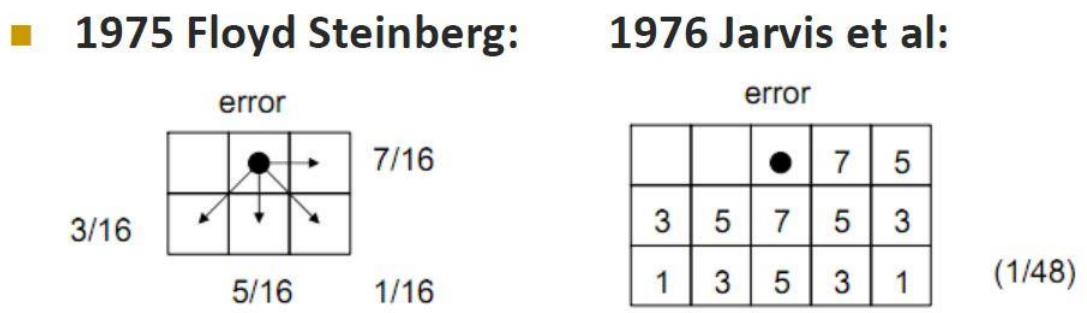
對比前一小題的結果，把 matrix size 調大確實有讓 dither 的效果突顯出來，整個圖的顆粒比較多也比較均勻。

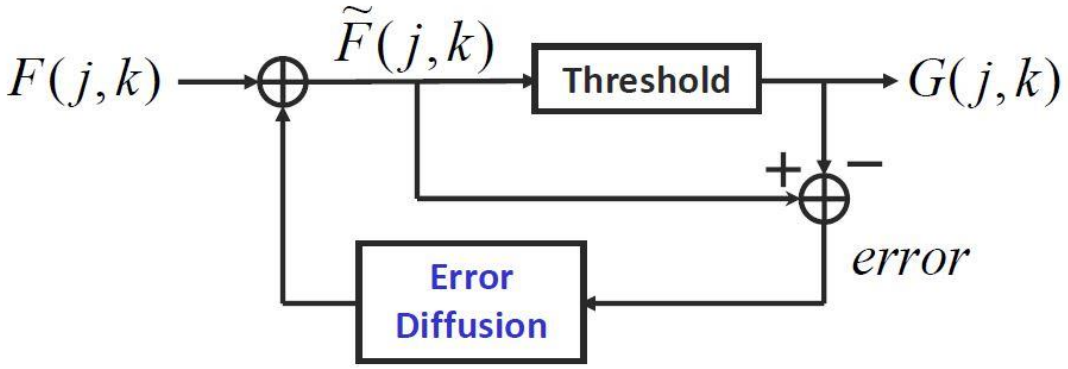
(c)

My motivation and approach:

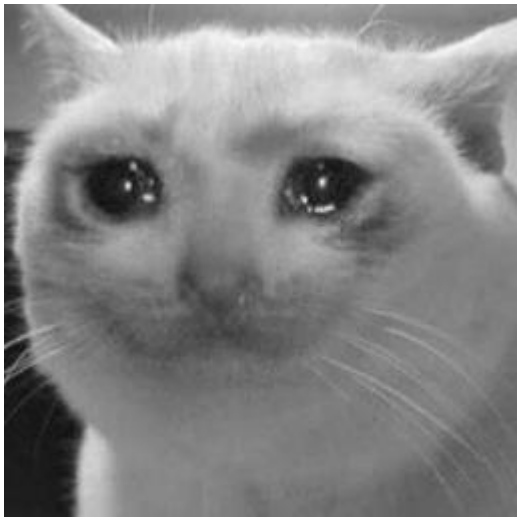
我選用的 error diffusion 方法為講義所介紹的 Floyd Steinberg 和 Jarvis et al，他們的 filter mask 如下圖。而 error diffusion 的做法為先將圖片的 pixel 跟 threshold(使用 128)比大小，轉成黑白後，根據 filter mask 去跟周圍的 pixel 做計算，把新舊之間的差值乘以 filter mask，再加上 pixel 值即為所求。

(Floyd Steinberg 實作參考這個網站:
https://stackoverflow.com/questions/55874902/floyd-steinberg-implementation-python)



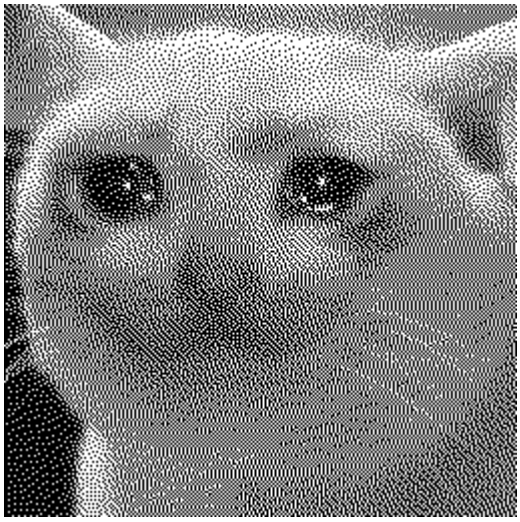


Original images:

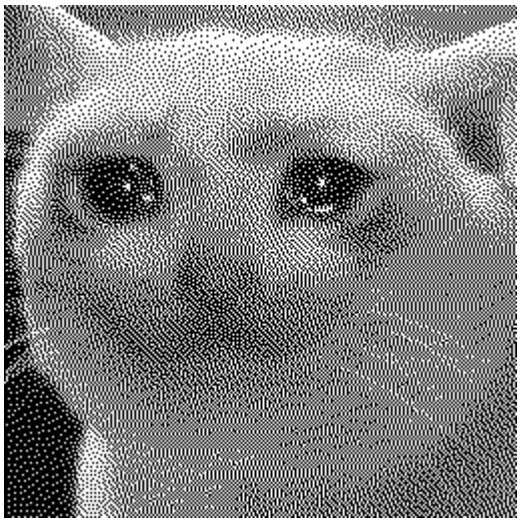


Output images:

Floyd Steinberg:



Jarvis et al:



Discussion of results:

兩者看起來幾乎差不多。不過Floyd Steinberg幾乎整張圖沒有空隙；而Jarvis et al會在某些地方比較空沒有顆粒，以此圖來說就是右半邊貓的臉，有些許空下來的地方。

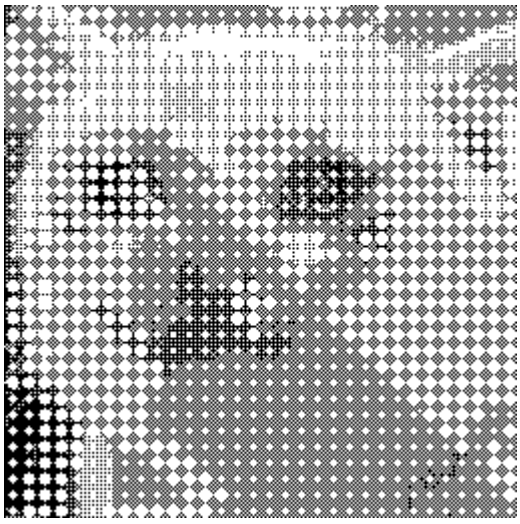
(d)

My motivation and approach:

將圖片分成32 * 32個8 * 8的區塊，每個區塊裡去計算白色pixel的數量，最後根據白色pixel的數量作為點大小的依據(半徑採用白色pixel的數量開根號除以2，是因為如果區塊內全為白色pixel，則半徑剛好是4)。

Original images:

Output images:



Discussion of results:

跟預期的圖片看起來有些差，但還是有做到把點放上去。

Problem 2: FREQUENCY DOMAIN

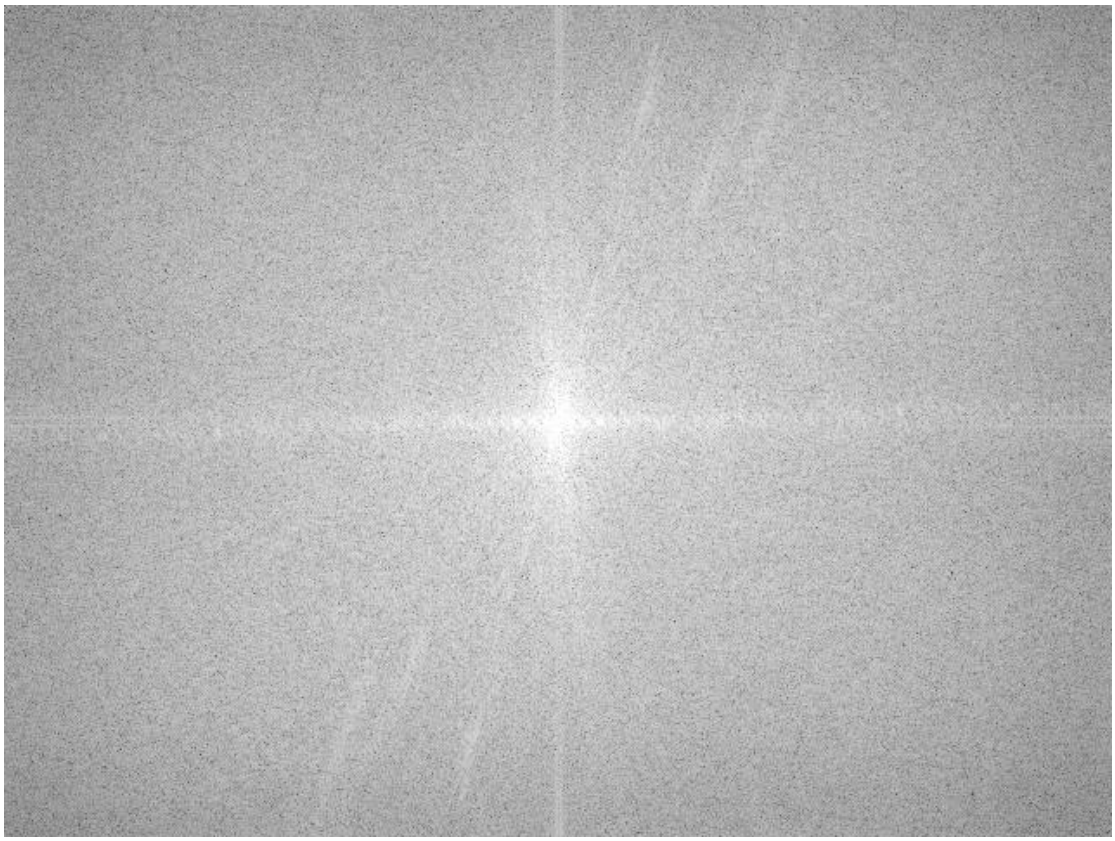
(a)

My motivation and approach:

使用 numpy 的函式 numpy.fft.fft2、numpy.fft.fftshift 來得到 magnitude，之後再將 magnitude 取絕對值取 log 乘以 20。

Original images:

Output images:



Discussion of results:

就是 frequency spectrum 的圖。

(b)

My motivation and approach:

根據(a)的結果，把 frequency spectrum 外圍高頻率的地方過濾掉，再用 numpy.fft.ifftshift、numpy.fft.ifft2，就是經過在 frequency domain 經過 low-pass filter 的結果。至於在 pixel domain 經過 low-pass filter 的結果，則可以使用作業 1 所寫過的 Gaussian filter。

Original images:

Output images:

frequency domain:

pixel domain:

Discussion of results:

pixel domain 的效果看起來不明顯；frequency domain 則是我將參數調整後，把 frequency spectrum 外圍從邊界數來 200 pixel 範圍的頻率全濾掉才比較顯著。

(c)

My motion and approach:

方法與(b)類似，只是變成把 frequency spectrum 中間低頻率的地方過濾掉。至於在 pixel domain 經過 high-pass filter 的結果，我在這邊使用知名的

$$\text{Laplacian filter}\left(\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right)。$$

Original images:

Output images:

frequency domain:

pixel domain:

Discussion of results:

pixel domain 的效果很明顯，物件的輪廓清楚；frequency domain 則是我權衡亮度與輪廓後，把 frequency spectrum 中間 5 * 5 pixel 範圍的頻率濾掉(濾掉太多圖會太黑)。

(d)

My motivation and approach:

跟(a)一樣做法，得到 frequency spectrum 的圖。

Original images:

Output images:

Discussion of results:

從 frequency spectrum 的圖可以看出，在向外到高頻率地方的時候一值有某些特殊的規則紋路，可見圖片應該存在某種固定模式的雜訊，我認為那應該就是圖片上的條紋所導致的。

(e)

My motivation and approach:

由於我們要去除雜訊，所以用 low-pass filter，做法如同(b)，把 frequency spectrum 外圍高頻率的地方過濾掉，再用 numpy.fft.ifftshift、numpy.fft.ifft2。

Original images:

Output images:

Discussion of results:

把 frequency spectrum 外圍從邊界數來 190 pixel 範圍的頻率全濾掉後，那些規則條紋才消失，但圖片變得很糊，不過至少有達成目標。