Compressione di immagini con perdita

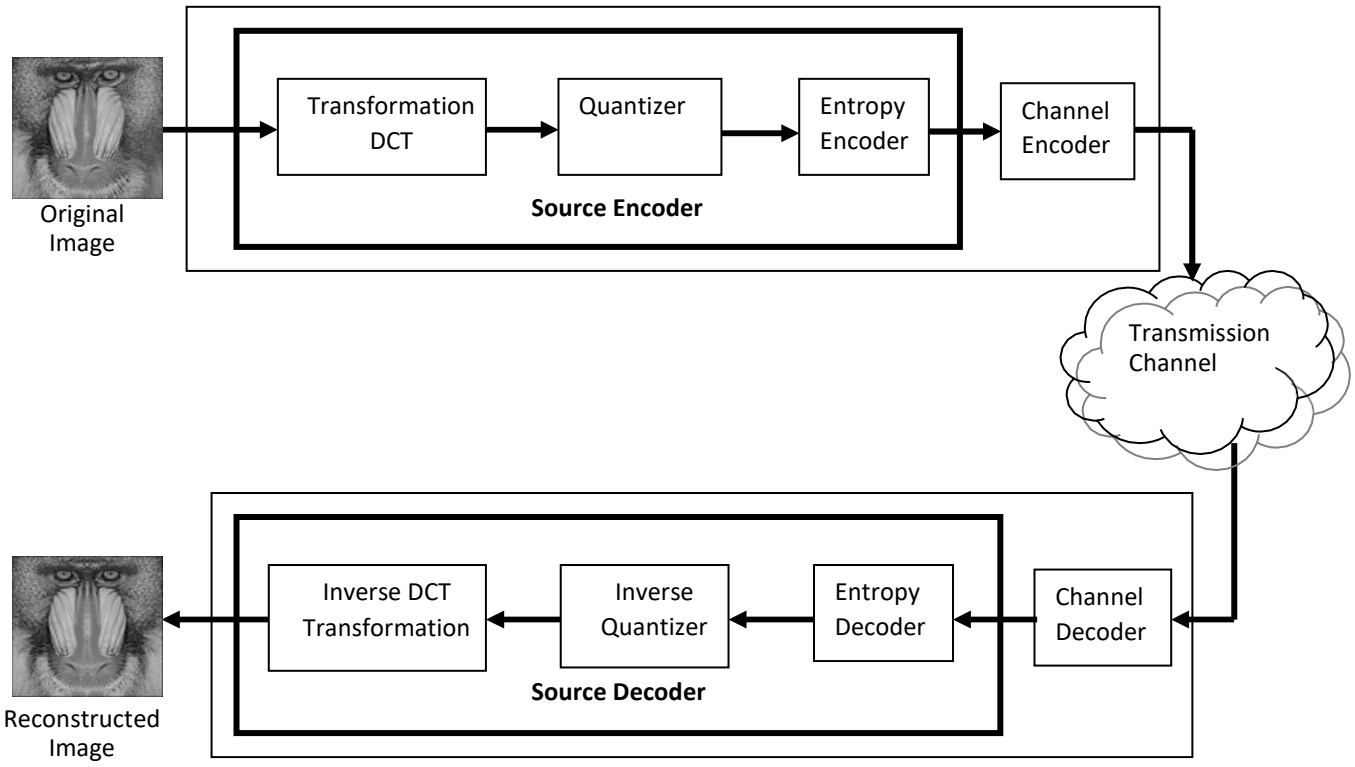
Introduzione

Esistono due tipi di compressione delle immagini, vale a dire: compressione senza perdita e compressione con perdita. Negli schemi di compressione senza perdita di dati, l'immagine ricostruita è la replica esatta dell'immagine originale. Mentre nella compressione di immagini con perdita, l'immagine ricostruita contiene un degrado rispetto all'originale a causa della perdita di informazioni.
L'obiettivo di base della compressione dell'immagine è ridurre il numero di bit richiesti dall'immagine per la sua memorizzazione senza che essa risulti troppo distorta dall'immagine originale. Ogni immagine è composta da due ingredienti principali, informazioni e ridondanza. La compressione delle immagini si ottiene rimuovendo le ridondanze di dati ottenendo una riduzione delle dimensioni dell'immagine al costo di una qualità inferiore.
Discuteremo in dettaglio la compressione delle immagini basata sulla codifica JPEG. Il processo JPEG è una forma ampiamente usata di compressione delle immagini con perdita che ruota attorno alla Trasformazione discreta del Coseno. Il DCT funziona separando le immagini in parti di frequenze diverse. Durante una fase chiamata quantizzazione, dove effettivamente si verifica una parte della compressione, le frequenze meno importanti vengono scartate, da qui l'uso del termine "lossy". Quindi, solo le frequenze più importanti rimaste vengono utilizzate per recuperare l'immagine nel processo di decompressione. Come un risultato, le immagini ricostruite contengono qualche distorsione; ma come vedremo presto, questi livelli di distorsione possono essere regolati durante la fase di compressione. Il metodo JPEG viene utilizzato sia per le immagini a colori che per quelle in bianco e nero, ma il focus di questo articolo sarà sulla compressione di quest'ultimo.

Il processo

Di seguito è riportata una panoramica generale del processo JPEG i quali punti verranno approfonditi in seguito.

1. L'immagine è suddivisa in blocchi di 8x8 pixel
2. Operando da sinistra a destra, dall'alto verso il basso, a ciascun blocco viene applicato la DCT
3. Ogni blocco viene compresso tramite la quantizzazione
4. L'array di blocchi compressi che costituiscono l'immagine viene archiviato in una quantità di spazio drasticamente ridotta
5. Se lo si desidera, l'immagine viene ricostruita attraverso la decompressione, un processo che utilizza l'Inversa Discrete Cosine Transform (IDCT)



Trasformata di Coseni Discreta (DCT):

La definizione DCT più comune di una sequenza 1-D di lunghezza N è chiamate DCT-II ed è la seguente:

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) cos\left[\frac{\pi(2x+1)u}{2N}\right] \qquad dove\ \alpha(u) = \begin{cases} \sqrt{1/N}, & se\ u = 0 \\ \sqrt{2/N}, & se\ u \neq 0 \end{cases}$$

È possibile anche definire la DCT in 2-D come diretta estensione di quella in 1-D nel modo seguente:

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos\left[\frac{\pi(2x+1)u}{2N}\right] cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

Utilizzata per processare immagini, e questa sarà la formula su cui ci concentreremo maggiormente.

Proprietà:

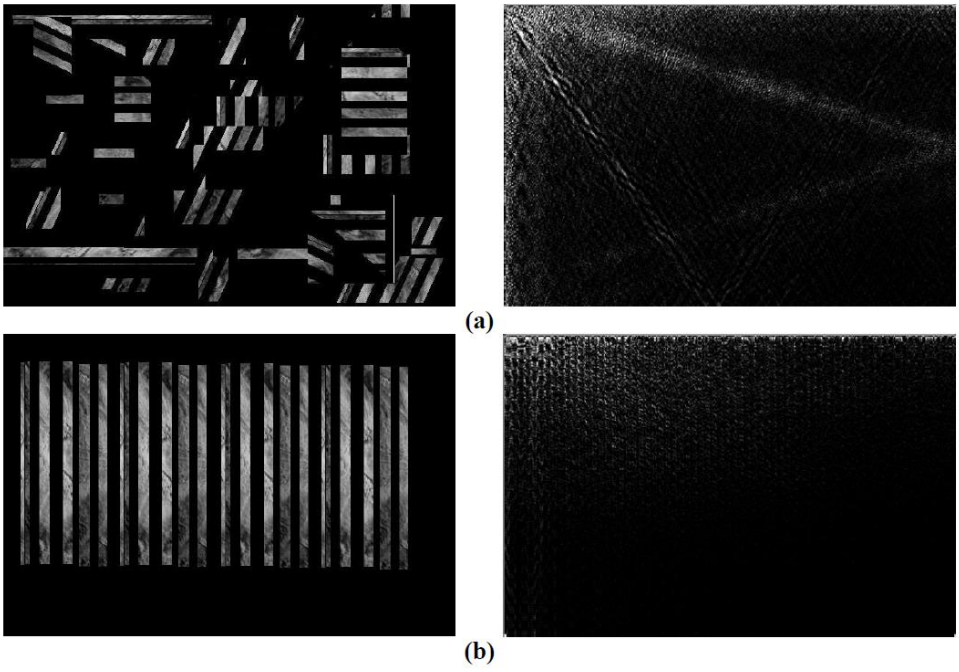
Questa sezione delinea alcune proprietà della DCT in 2-D che hanno un particolare valore per le applicazioni di elaborazione delle immagini.

Decorrelazione:

Come discusso in precedenza, il principale vantaggio della trasformazione dell'immagine è la rimozione della ridondanza tra pixel vicini, cioè la somiglianza tra pixel vicini. Ciò porta a coefficienti di trasformazione non correlati che possono essere codificati in modo indipendente. Tutto ciò mira a de-correlare i dati originali e a compattare una grande porzione dei dati dell'immagine in relativamente pochi coefficienti di trasformazione.

Energy Compaction:

Se definiamo l'energia di una matrice come la somma dei quadrati dei valori abbiamo che la DCT, pur mantenendo invariata l'energia dell'input, la ridistribuisce in un numero inferiore di coefficienti. Ciò consente di scartare i coefficienti con ampiezze relativamente piccole senza introdurre distorsioni visive nell'immagine ricostruita. DCT esibisce un'eccellente compattazione energetica per immagini altamente correlate.



(a)

(b)

Consideriamo le 2 immagini a sinistra con la sua DCT accanto. L'immagine non correlata (a) presenta variazioni di intensità più nette rispetto all'immagine correlata(b). Pertanto, il primo ha più contenuto ad alta frequenza rispetto al secondo. Chiaramente, l'immagine non correlata ha la sua energia distribuita, mentre l'energia dell'immagine correlata è impacchettata nella regione di bassa frequenza (cioè, in alto a sinistra).

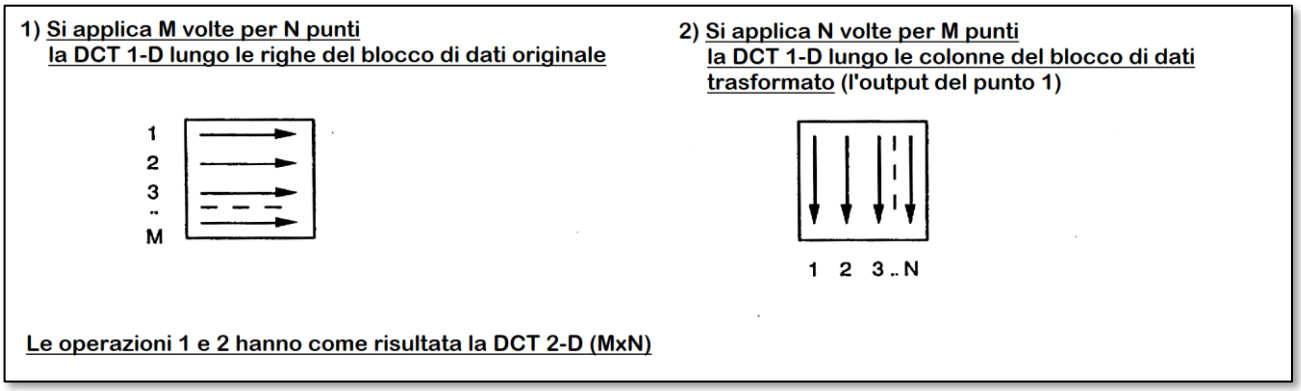
Separabilità:

Tutte le DCT multidimensionali possono essere scomposte in successive applicazioni della trasformata unidimensionale per ogni direzione. Nel nostro caso, per la DCT 2-D possiamo riscriverla come il prodotto di due trasformate unidimensionali, la prima che lavora sulle righe mentre la seconda sulle colonne, questa proprietà è detta separabilità ed è utile per ridurre la complessità dell'algoritmo.

Equazione riscritta separando il calcolo delle colonne da quello delle righe:

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\left(cos\left[\frac{\pi(2x+1)u}{2N}\right]\left(\sum_{y=0}^{N-1} f(x,y)\ cos\left[\frac{\pi(2y+1)v}{2N}\right]\right)\right)$$

Quindi un possibile algoritmo per la DCT 2-D dovrebbe prima applicare un algoritmo DCT 1-D alle righe del blocco di input, applicando successivamente all'output un algoritmo DCT 1-D alle colonne del blocco di dati trasformato. Le due applicazioni possono essere invertire e il risultato ottenuto sarà sempre lo stesso.



Simmetria:

Un altro sguardo alle operazioni di riga e colonna nella formula data grazie alla separabilità rivela che queste operazioni sono identiche. Tale trasformata è chiamata trasformata simmetrica. Una trasformata separabile e simmetrica può essere espressa nella forma

$$T = AfA',$$

dove A è una matrice di trasformazione simmetrica N × N con valori $a(i,j)$ ottenuti da

$$a(i,j) = \alpha(j)\sum_{j=0}^{N-1} cos\left[\frac{\pi(2j+1)i}{2N}\right],$$

e *f* è la matrice dell'immagine NxN. Questa proprietà è estremamente utile perché è possibile precalcolare la matrice di trasformazione e applicarla poi all'immagine, migliorando così l'efficienza dell'algoritmo.

Funzione inversa:

La funzione inversa della DCT in 2-D è descritta nel seguente modo:

$$f(x,y) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\alpha(u)\alpha(v)\ C(u,v)\ cos\left[\frac{\pi(2x+1)u}{2N}\right]cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

Ed è possibile definirla anche come moltiplicazioni di matrici. Ed essendo A una matrice ortonormale, la sua inversa è la stessa della sua trasposta(A^{-1}=*A*'). Pertanto, la DCT 2-D inversa può essere scritta come:

$$f = A'TA.$$

Ora che abbiamo discusso la teoria alla base della DCT possiamo con l'algoritmo alla base della codifica JPEG.

La DCT kernel Matrix

Grazie alle proprietà di separabilità e simmetria discusse precedentemente è possibile precalcolando la matrice di coseni, detta Kernel Matrix, utilizzando la seguente equazione:

$$A_{i,j} = \begin{cases} \frac{1}{\sqrt{N}} & if\ \ i = 0 \\ \sqrt{\frac{2}{N}} cos\left[\frac{(2j+1)i\pi}{2N}\right] & if\ \ i > 0 \end{cases}$$

E per un blocco 8x8 la matrice risultante è:

$$A = \begin{bmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4157 & .2778 & .0975 & -.0975 & -.2778 & -.4157 & -.4904 \\ .4619 & .1913 & -.1913 & -.4619 & -.4619 & -.1913 & .1913 & .4619 \\ .4157 & -.0975 & -.4904 & -.2778 & .2778 & .4904 & .0975 & -.4157 \\ .3536 & -.3536 & -.3536 & .3536 & .3536 & -.3536 & -.3536 & .3536 \\ .2778 & -.4904 & .0975 & .4157 & -.4157 & -.0975 & .4904 & -.2778 \\ .1913 & -.4619 & .4619 & -.1913 & -.1913 & .4619 & -.4619 & .1913 \\ .0975 & -.2778 & .4157 & -.4904 & .4904 & -.4157 & .2778 & -.0975 \end{bmatrix}$$

La prima riga della matrice ha tutte le voci pari a $^1/_{\sqrt{8}}$ come previsto dall'equazione.

Le colonne di A formano un insieme ortogonale, quindi A è una matrice ortogonale come già detto in precedenza. Quando si fa l'inversa della DCT l'ortogonalità di T è importante, poiché l'inverso di T è T' che è facile da calcolare.

Applicare la DCT a un blocco 8x8

Innanzitutto, visto che stiamo lavorando su immagini in bianco e nero, la scala di grigi per ogni pixel viene rappresentata da un numero intero tra 0 e 255. Siccome le immagini sono composte da migliaia blocchi di 8x8 pixel, considereremo solo l'applicazione della codifica JPEG ad un solo blocco, il procedimento è uguale per tutti gli altri nell'ordine specificato in precedenza.

Prendiamo in esempio un blocco di un'immagine:

$$Originale = \begin{bmatrix} 115 & 112 & 112 & 112 & 108 & 100 & 96 & 98 \\ 113 & 110 & 109 & 110 & 107 & 100 & 97 & 99 \\ 100 & 98 & 97 & 98 & 96 & 92 & 90 & 92 \\ 86 & 84 & 83 & 85 & 85 & 84 & 85 & 87 \\ 86 & 84 & 84 & 86 & 89 & 91 & 93 & 94 \\ 98 & 97 & 98 & 100 & 103 & 107 & 108 & 109 \\ 116 & 115 & 115 & 117 & 121 & 124 & 125 & 124 \\ 131 & 130 & 130 & 131 & 135 & 137 & 137 & 135 \end{bmatrix}$$

Siccome la DCT è progettata per lavorare in un range tra -128 e 127, il blocco originale viene "livellato" sottraendo 128 ad ogni casella, ottenendo così la seguente matrice:

$$f = \begin{bmatrix} -13 & -16 & -16 & -16 & -20 & 28 & -32 & -30 \\ -15 & -18 & -19 & -18 & -21 & -28 & -31 & -29 \\ -28 & -30 & -31 & -30 & -32 & -36 & -38 & -36 \\ -42 & -44 & -45 & -43 & -43 & -44 & -43 & -41 \\ -42 & -44 & -44 & -42 & -39 & -37 & -35 & -34 \\ -30 & -31 & -31 & -28 & -25 & -21 & -20 & -19 \\ -12 & -13 & -13 & -11 & -7 & -4 & -3 & -4 \\ 3 & 2 & 2 & 3 & 7 & 9 & 9 & 7 \end{bmatrix}$$

Ora possiamo applicare la trasformata discreta di coseni, che si ottiene mediante la moltiplicazione delle matrici nel seguente modo: $T = AfA'$. Perciò la matrice *f* viene prima moltiplicata a sinistra per la matrice DCT *A* trasformando le file. Le colonne vengono successivamente trasformate moltiplicando la matrice risultante alla trasposta della matrice DCT. Alla fine si ottiene la seguente matrice:

$$T = \begin{bmatrix} -187.9 & 0.1 & -0.1 & 2.9 & 5.4 & -0.1 & 0.2 & 0.0 \\ -59.9 & 30.5 & -4.5 & -5.0 & 5.6 & -0.5 & 0.2 & -0.2 \\ 101.9 & 12.0 & -4.4 & 0.2 & 0.3 & -0.3 & 0.3 & 0.6 \\ -15.0 & -4.9 & 0.5 & 0.1 & -0.1 & 0.2 & 0.4 & -0.2 \\ -9.9 & 0.2 & 0.4 & -0.1 & -0.1 & 0.2 & 0.0 & 0.6 \\ -7.5 & 0.2 & 0.1 & -0.5 & -0.1 & 0.1 & -0.1 & -0.2 \\ 0.1 & 0.5 & -0.5 & -0.3 & -0.3 & -0.2 & 0.4 & -0.4 \\ 0.3 & 0.1 & -0.2 & 0.3 & -0.4 & 0.0 & 0.1 & 0.4 \end{bmatrix}$$

Quantizzazione:

Il blocco 8x8 di coefficienti DCT è ora pronto per la compressione per quantizzazione. Una caratteristica notevole e molto utile del processo JPEG è che in questo passaggio, si possono ottenere vari livelli di compressione e qualità delle immagini attraverso la selezione di matrici di quantizzazione specifiche. Ciò consente all'utente di decidere livelli di qualità compresi tra 1 e 100, dove 1 fornisce la qualità dell'immagine più scadente e la massima compressione, mentre 100 offre la migliore qualità e la compressione più bassa. Di conseguenza, il rapporto qualità/compressione può essere personalizzato per soddisfare le diverse esigenze.
Esperimenti soggettivi che coinvolgono il sistema visivo umano hanno portato alla matrice di quantizzazione standard JPEG. Con un livello di qualità pari a 50, questa matrice rende elevati sia la compressione che la qualità dell'immagine decompressa.

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Se si desidera un altro livello di qualità e compressione, è possibile utilizzare multipli scalari della matrice di quantizzazione standard JPEG.
Per un livello di qualità superiore a 50 (inferiore compressione, qualità dell'immagine superiore), la matrice di quantizzazione standard viene moltiplicata per $\left(\frac{100-\text{livello di qualità}}{50}\right)$.
Per un livello di qualità inferiore a 50 (maggiore compressione, qualità dell'immagine inferiore), la matrice di quantizzazione standard viene moltiplicata per $\left(\frac{50}{\text{livello di qualità}}\right)$. Il ridimensionato la matrice di quantizzazione viene quindi arrotondata e ritagliata per avere valori interi positivi che vanno da 1 a 255.

$$Q_n = \begin{cases} \left(\frac{100-n}{50}\right) Q_{50} & se\ n > 50 \\ \left(\frac{50}{n}\right) Q_{50} & se\ n < 50 \end{cases}$$

Ad esempio, le seguenti matrici di quantizzazione producono livelli di qualità di 10 e 90

$$Q_{10}=\begin{bmatrix} 80 & 55 & 50 & 80 & 120 & 200 & 255 & 255 \\ 60 & 60 & 70 & 95 & 130 & 255 & 255 & 255 \\ 70 & 65 & 90 & 120 & 200 & 255 & 255 & 255 \\ 70 & 85 & 110 & 145 & 255 & 255 & 255 & 255 \\ 90 & 110 & 185 & 255 & 255 & 255 & 255 & 255 \\ 120 & 175 & 255 & 255 & 255 & 255 & 255 & 255 \\ 245 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix} \qquad Q_{90}=\begin{bmatrix} 3 & 2 & 2 & 3 & 4 & 8 & 10 & 12 \\ 2 & 2 & 2 & 3 & 5 & 11 & 12 & 11 \\ 2 & 2 & 3 & 4 & 8 & 11 & 13 & 11 \\ 2 & 3 & 4 & 5 & 10 & 17 & 16 & 12 \\ 3 & 4 & 7 & 11 & 13 & 21 & 20 & 15 \\ 4 & 7 & 11 & 12 & 16 & 20 & 22 & 18 \\ 9 & 12 & 15 & 17 & 20 & 24 & 24 & 20 \\ 14 & 18 & 19 & 19 & 22 & 20 & 20 & 19 \end{bmatrix}$$

La quantizzazione si ottiene dividendo ciascun elemento del blocco dell'immagine trasformata T per l'elemento corrispondente nella matrice di quantizzazione e quindi arrotondando al più vicino valore intero. Per il passaggio seguente, viene utilizzata la matrice di quantizzazione Q50.

$$C_{i,j} = round\left(\frac{T_{i,j}}{Q_{i,j}}\right) \qquad \rightarrow \qquad C = \begin{bmatrix} -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ricordiamo che i coefficienti situati vicino all'angolo in alto a sinistra corrispondono alle frequenze più basse - a cui l'occhio umano è più sensibile - del blocco di immagini, infatti i coefficienti della matrice di quantizzazione sono scelti in modo da preservare le frequenze più basse e azzerare quelle alte. Infatti, gli zeri rappresentano le frequenze meno importanti e più elevate che sono state scartate, dando origine alla parte con perdita di compressione. Come accennato in precedenza, per ricostruire l'immagine verranno utilizzati solo i restanti coefficienti diversi da zero.

Codifica di Huffman:

Prima di proseguire con l'ultima fase, approfondiamo cos'è la codifica di Huffman la quale verrà utilizzata per comprimere ulteriormente i nostri dati.

Nella teoria dell'informazione, per *Codifica di Huffman* si intende un algoritmo di codifica *lossless* dei simboli usato per la compressione di dati, basato sul principio di trovare il sistema ottimale per codificare stringhe basato sulla frequenza relativa di ciascun carattere. La codifica di Huffman effettua un'analisi statistica del dato da comprimere, da cui inizia una fase di analisi per decidere i codici da assegnare ad ogni simbolo in base alla loro frequenza.
Il guadagno di spazio al termine della compressione è dovuto al fatto che gli elementi che si ripetono frequentemente sono identificati da un codice breve, che occupa meno spazio di quanto ne occuperebbe la loro codifica normale. Viceversa, gli elementi rari nel file originale ricevono nel file compresso una codifica lunga, che può richiedere, per ciascuno di essi, uno spazio anche notevolmente maggiore di quello occupato nel file non compresso.

Algoritmo Codifica di Huffman. *Sia X un insieme di n simboli* $x_1, x_2, \ldots, x_n$ *di probabilità* $p_1, p_2, \ldots, p_n$ *con* $\sum_i p_i = 1$. *Creiamo un nodo-foglia per ogni simbolo e aggiungiamolo a una coda in cui gli elementi con probabilità minore hanno priorità più alta.*

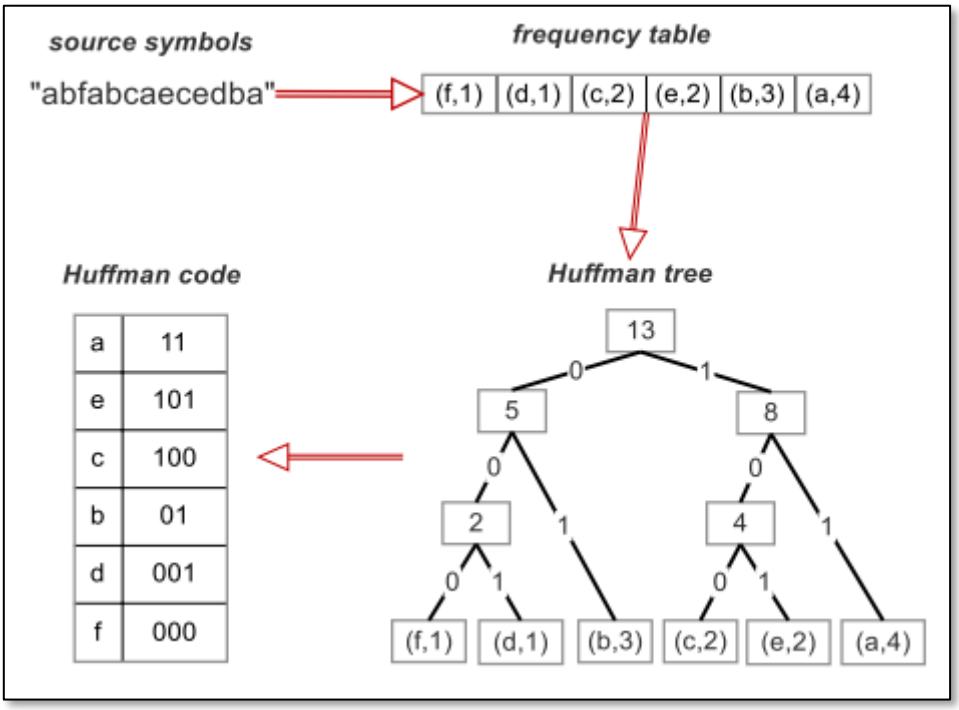
Fino a che la coda contiene più di un elemento:

1. *rimuoviamo i due nodi a più alta priorità (probabilità più bassa) dalla coda*
2. *creiamo un nodo interno con i due nodi come figli e probabilità pari alla somma delle probabilità*
3. *associamo il simbolo 0 all'arco verso il figlio di probabilità minore e 1 all'altro*
4. *aggiungiamo il nuovo nodo alla coda nel rispetto della priorità*

Il nodo rimanente è la radice, l'albero è completo e la codifica di ogni simbolo si ottiene concatenando i bit associati agli archi del cammino dalla radice alla foglia corrispondente.

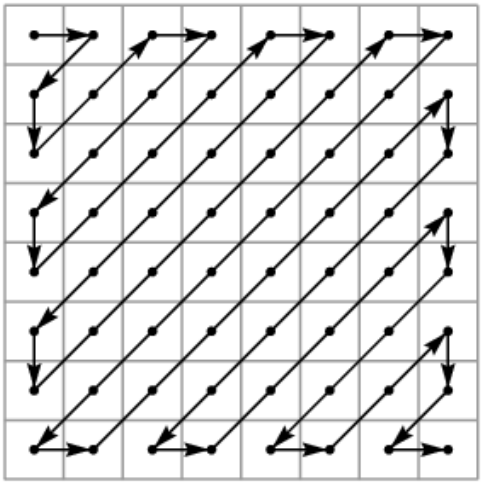
La codifica di Huffman da 2 proprietà importanti:

- È una codifica istantanea, cioè se nessun codice che rappresenta un simbolo è prefisso di un altro codice. Questo è importante perché una codifica istantanea significa efficienza.
- La codifica di Huffman è ottimale, nessun'altra codifica istantanea senza perdita di informazioni può garantire una compressione media migliore.



Codifica Entropica:

La matrice quantizzata C è ora pronta per il passaggio finale di compressione. Prima della memorizzazione, tutti i coefficienti di C vengono convertiti da un encoder in un flusso di dati binari (01101011 ...). Dopo la quantizzazione, è abbastanza comune che la maggior parte dei coefficienti sia uguale a zero, grazie al quale è possibile ridurre nettamente lo spazio necessario per rappresentare la matrice. Il processo di codifica è il seguente:



- Tutti i coefficienti quantizzati sono ordinati in una sequenza a "zig-zag". Questo ordine facilita la successiva fase ponendo i coefficienti di bassa frequenza (più probabilmente non nulli) prima di quelli di alta frequenza. Quindi la nostra matrice si trasformerebbe nel seguente vettore:

  $$[12, 0, -5, 7, 3, 0, 0, 0, 1, -1, -1, 0, 0, 0, \dots, 0, 0, 0]$$

  Il primo coefficiente della matrice (coordinate 0, 0) è chiamato coefficiente DC, mentre i restanti 63 coefficienti sono chiamati coefficienti AC.

- Dopo aver ordinato i coefficienti, è possibile codificarli con una codifica di tipo *run-lenght encoding* (RLE), in cui verranno codificati in modo efficiente le sequenze di zeri. La sequenza "zig-zag" di coefficienti della matrice quantizzata viene quindi convertita in una sequenza intermedia di simboli nella seguente maniera:
  - Nella sequenza intermedia di simboli, ogni coefficiente AC è rappresentato da una coppia di simboli:
    - Simbolo-1 (RUNLENGTH, SIZE)
    - Simbolo-2 (AMPLITUDE)

    Dove RUNLENGTH rappresenta il numero di valori nulli (gli zeri) che precedono un valore non nullo; SIZE indica il numero di bit necessari a codificare l'ampiezza (AMPLITUDE).
  - Il coefficiente DC del blocco è codificato invece dalla coppia:
    - Simbolo-1 (SIZE)
    - Simbolo-2 (AMPLITUDE)

  La sequenza di zeri che termina senza un numero diverso da zero non viene codificata in quanto il numero di elementi finali, nella decodifica, dovrà comunque essere uguale a 63, quindi una volta decodificati gli elementi sino al primo non nullo, aggiungendo degli zeri per completare la sequenza sino a 63 elementi. Si usa il simbolo (0, 0) per tenere traccia di dove finisce un blocco e codificare il fatto che i coefficienti restanti sono tutti nulli.
  Quindi la nostra sequenza si trasformerebbe in:

  $$(4)(12), (1,3)(-5), (0,3)(7), (3,1)(1), (3,1)(-1), (3,1)(-1), (0,0)$$

- Infine, la sequenza intermedia è codificata in una sequenza binaria mediante l'uso di Huffman. Le tabelle di codifica di Huffman, in cui viene associato ad ogni simbolo della sequenza un codice di Huffman, possono essere calcolate prima della fase di compressione, attraverso un'analisi statistica dei dati da rappresentare oppure si possono usare delle tabelle di default già stabilite a priori. Lo standard JPEG mette a disposizione delle tabelle definite a priori che sono state stabilite attraverso l'analisi di immagini con caratteristiche frequenti. Grazie a questo si otterrà una ulteriore compressione grazie al fatto che i simboli con frequentemente maggiore vengono rappresentati con sequenze di bit corte, quelli meno frequenti con sequenze più lunghe.
  Ovviamente queste tabelle dovranno essere successivamente memorizzate per poter permettere successivamente la decodifica dell'immagine.

Decompressione:

Ovviamente la fase “ricostruzione” di un’immagine segue alla rovescia i passaggi descritti prima.
Quindi partendo da uno stream di bit si ricrea la nostra matrice di elementi quantizzati grazie alle tabelle di Huffman che ci permettono di creare la sequenza intermedia. Da questa si costruisce il vettore di lunghezza 64 e applicando alla rovescia l’algoritmo “zig-zag” riotteniamo la matrice C 8x8.
A questo punto ogni elemento di C viene moltiplicato per l’elemento corrispondente nella matrice di quantizzazione originariamente utilizzata.

$$R_{i,j} = Q_{i,j} \mathrm{x} C_{i,j}$$

$$R = \begin{bmatrix} -192 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -60 & 36 & 0 & 0 & 0 & 0 & 0 & 0 \\ 98 & 13 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

L'IDCT viene quindi applicato alla matrice R, che viene arrotondata al numero intero più vicino. Infine, 128 viene aggiunto a ciascun elemento di quel risultato, dandoci la versione decompressa JPEG N del blocco dell’immagine 8x8 originale.

$$N = round(T' R\, T) + 128$$

Confronto delle matrici:

$$Decompressa = \begin{bmatrix} 111 & 103 & 101 & 99 & 99 & 101 & 107 & 117 \\ 102 & 92 & 89 & 88 & 89 & 93 & 102 & 114 \\ 93 & 91 & 89 & 88 & 89 & 93 & 102 & 114 \\ 93 & 79 & 81 & 84 & 90 & 97 & 106 & 119 \\ 89 & 75 & 82 & 91 & 101 & 111 & 118 & 124 \\ 118 & 115 & 119 & 124 & 129 & 133 & 133 & 131 \\ 139 & 144 & 146 & 147 & 147 & 145 & 141 & 135 \\ 139 & 144 & 144 & 144 & 146 & 140 & 137 & 133 \end{bmatrix}$$

$$Originale = \begin{bmatrix} 115 & 112 & 112 & 112 & 108 & 100 & 96 & 98 \\ 113 & 110 & 109 & 110 & 107 & 100 & 97 & 99 \\ 100 & 98 & 97 & 98 & 96 & 92 & 90 & 92 \\ 86 & 84 & 83 & 85 & 85 & 84 & 85 & 87 \\ 86 & 84 & 84 & 86 & 89 & 91 & 93 & 94 \\ 98 & 97 & 98 & 100 & 103 & 107 & 108 & 109 \\ 116 & 115 & 115 & 117 & 121 & 124 & 125 & 124 \\ 131 & 130 & 130 & 131 & 135 & 137 & 137 & 135 \end{bmatrix}$$

$$(Originale - Decompressa) = \begin{bmatrix} 4 & 9 & 11 & 13 & 9 & -1 & -11 & -19 \\ 11 & 19 & 20 & 22 & 18 & 7 & -5 & -15 \\ 7 & 19 & 16 & 14 & 6 & -5 & -16 & -25 \\ 5 & 20 & 14 & 7 & -3 & -15 & -24 & -32 \\ -3 & 9 & 2 & -5 & -12 & -20 & -25 & -30 \\ -20 & -18 & -22 & -24 & -26 & -26 & -25 & -22 \\ -23 & -29 & -31 & -30 & -26 & -21 & -16 & -11 \\ -8 & -14 & -14 & -16 & -8 & -3 & 0 & 2 \end{bmatrix}$$

Questo è un risultato notevole, considerando che più dell'80% dei coefficienti DCT è stato scartato, perché era diventato un valore nullo, prima della decompressione / ricostruzione del blocco immagine. Essendo che risultati simili si otterranno per il resto dei blocchi dell'immagine il risultato finale sarà qualcosa di molto simile all'originale essendo che ci sono 256 scale di grigio in un'immagine in bianco e nero e differenze del genere sono appena percettibili dell'occhio umano. E questo utilizzando la matrice di quantizzazione Q_{50}, è possibile quindi avere una maggiore qualità sacrificando un po' di compressione.