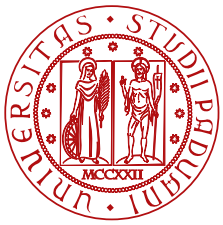


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Analisi comparativa di tecniche di compressione di immagini con reti neurali

Relatore
Prof. Cagnazzo Marco

Laureando
Roncolato Francesco
Matricola: 2032442

ANNO ACCADEMICO 2023-2024

Data di laurea 27/09/2024

Abstract

Fruizione e condivisione di contenuti multimediali sono ormai un aspetto chiave dell'umanità e per questo è necessario avere un sistema efficiente. La compressione ricopre un ruolo importante nell'era di Internet e porta con sé molteplici implicazioni pratiche ed economiche.

In questa tesi si presentano i concetti di base dell'elaborazione di contenuti digitali e della compressione. Si analizzano le metriche per valutare un sistema di compressione e le tecniche tradizionali più conosciute: JPEG e JPEG2000. Infine si mostra con esempi pratici come l'intelligenza artificiale permei anche in questo campo, affermandosi con risultati convincenti ma non ancora rivoluzionari.

Sommario

Nel primo capitolo una breve introduzione sull'importanza della compressione nell'era moderna e a seguire alcuni concetti di base dell'elaborazione e compressione delle immagini utili per comprendere al meglio la restante parte dell'elaborato. Si mostrano il concetto di immagine digitale, la differenza tra codifiche lossy e lossless e la conversione analogico-digitale e a finire il concetto di spazio di colori.
Nel secondo capitolo si illustrano alcune metriche rilevanti per valutare un sistema di compressione. In particolare si definisce il concetto di compression ratio, alcune tra le più note metriche oggettive per la qualità e per la computazione e infine le metriche soggettive.
Nel terzo capitolo si mostra lo stato dell'arte delle tecniche tradizionali più diffuse per la compressione lossy in termini di funzionamento e punti di forza e di debolezza.
Nel quarto capitolo si mostrano i concetti di base delle reti neurali, la suddivisione in famiglie di reti e alcuni esempi di applicazioni nell'ambito della compressione delle immagini.
Nel quinto capitolo l'analisi di una più recente tecnica di compressione che usa le reti neurali e la discussione dei risultati ottenuti, confrontando quanto riportato nell'articolo con quanto misurato personalmente.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 L'avvento di Internet

Gli eventi legati all'invenzione e allo sviluppo di Internet della fine del ventesimo secolo hanno sancito una svolta storica nella società. La globalizzazione della comunicazione ha eliminato le distanze geografiche favorendo lo scambio culturale e ridefinendo le dinamiche sociali. L'informazione scientifica, culturale, storica e di attualità è diventata di immediato accesso al pubblico, favorendo una diffusione più rapida e capillare del sapere e permettendo a tutti una maggiore partecipazioni alle dinamiche mondiali. L'esplosione dell'economia digitale ha portato le cosiddette "Big Tech" come Apple, Microsoft, NVIDIA, Alphabet (Google), Amazon e Meta ai primi posti tra le aziende con la più alta capitalizzazione [1][2] e ad affermarsi come pilastri dell'economia globale grazie alle loro capacità di sfruttare la rete per connettere persone e mercati. Grazie a piattaforme come YouTube, Facebook, Instagram e TikTok chiunque può esprimere le proprie idee o diventare un content creator oppure usufruire gratuitamente dei contenuti altrui rivoluzionando il ruolo sociale della creatività e dell'intrattenimento.

Fin da quel momento, e in modo più importante dai primi anni 2000, il numero totale di utenti globali di Internet è in crescita. In particolare negli ultimi anni si sta registrando un aumento non trascurabile nei paesi in via di sviluppo e in quelli meno sviluppati [3]. Secondo le proiezioni di Cisco [4], dal 2018 al 2023 gli utenti Internet sono aumentati da 3.9 a 5.3 miliardi passando da una media di 2.4 dispositivi pro capite a 3.6. Questa crescita, unita alla diffusione di nuove tecnologie ad alto consumo come i servizi di streaming ad alta e altissima definizione, i sistemi di AR/VR o IoT, porta inevitabilmente ad un aumento del traffico dati. Secondo le stime di Analysys Mason [5] infatti, il traffico dati mondiale triplicherà tra il 2022 e il 2028. In questo contesto di espansione digitale, la compressione ricopre un ruolo sempre più importante.

1.2 Principi generali della compressione dei dati

La compressione dei dati è un processo che consiste nel codificare o rappresentare l'informazione utilizzando un numero minore di bit. Senza le tecniche di compressione, la mole di traffico che circola in rete al giorno d'oggi non sarebbe sostenibile dalle infrastrutture. La capacità di comprimere i dati permette di risparmiare grandi quantità di spazio nella memoria locale o sui servizi cloud oltre a migliorare le performance in termini di tempo di caricamento e di trasmissione del contenuto a parità di bitrate (tasso di trasmissione). Tutti questi aspetti concorrono nell'aumentare la *Quality of Experience* o "qualità dell'esperienza" (QoE) limitando i tempi di attesa e permettono risparmi notevoli ai costruttori di dispositivi e infrastrutture.

Indipendentemente dalla natura del dato trattato, si possono distinguere due macro tipologie di compressione: *lossless* e *lossy*. La codifica lossless o "senza perdita" si basa su tecniche che permettono di ricostruire perfettamente, ovvero senza errori, il dato iniziale a partire dalla versione compressa. La codifica lossy o "con perdita" invece, ammette che il dato ricostruito presenti delle differenze rispetto a quello originale. Naturalmente l'obiettivo della compressione non è quello di corrompere l'informazione. Per questo motivo le tecniche lossy cercano di minimizzare l'errore commesso, anche sfruttando le debolezze e le caratteristiche della percezione umana. Ad esempio si può decidere di "sbagliare" un po' nella rappresentazione del colore e mantenere una precisione più alta nella luminanza, dato che l'occhio umano è più sensibile a quest'ultima (vedi 1.3.2).

Sebbene possa sembrare strano decidere di manipolare dei dati fino a corromperli parzialmente, le tecniche lossy sono molto utilizzate in quanto riescono a comprimere molto di più il dato. La compressione lossless è sostanzialmente limitata dal primo teorema di Shannon [6] mentre quella lossy può essere spinta molto oltre, pagando in termini di qualità ottenuta.

1.3 Immagini digitali

Nel seguito della trattazione si affronteranno vari aspetti della compressione di immagini digitali. Qui, brevemente, un'introduzione a questo tipo di contenuto multimediale.

1.3.1 Digitalizzazione

Un segnale analogico è una funzione che dipende da una variabile continua ed assume valori continui ad esempio la luce che entra nell'obiettivo per la registrazione di un video. All'interno di un sistema informatico però, l'elaborazione (e quindi anche la compressione) può avvenire solamente su segnali digitali ovvero funzioni con dominio e codominio discreti e finiti. Il codominio di un segnale digitale è detto anche alfabeto e i suoi valori devono essere rappresentabili

come stringhe di bit. È quindi necessario il processo di *digitalizzazione* o *Analog-to-Digital Conversion* (ADC) che consiste principalmente di due fasi: campionamento e quantizzazione. Sebbene esista il processo contrario (*Digital-to-Analog Conversion* o DAC), la digitalizzazione è un processo irreversibile poiché intrinsecamente con perdita.

Si immagini di voler registrare un video. La luce che entra nell'obiettivo della videocamera può essere considerata un segnale analogico. Il primo passo per rendere possibile la memorizzazione di un segnale analogico è il campionamento ovvero la discretizzazione della variabile indipendente. Questo passaggio permette di memorizzare su supporti fisici i segnali a tempo discreto ma ampiezza continua. Le pellicole cinematografiche sono un ottimo esempio di questo passaggio in quanto sono composte da un numero finito di fotogrammi, ciascuno dei quali è un'immagine analogica. Nelle foto e videocamere digitali, i sensori (solitamente di tipo CCD o CMOS) introducono il campionamento anche nella variabile dipendente, ovvero nelle coordinate spaziali di ciascun frame. È in questo modo che l'immagine analogica viene trasformata in una griglia di pixel, l'unità minima nelle immagini digitali. Chiaramente le immagini sono intrinsecamente tempo-discrete e quindi la loro cattura non richiede la fase di campionamento.

Il secondo e ultimo passo della digitalizzazione è la quantizzazione ovvero la discretizzazione della variabile dipendente. Nel caso delle immagini digitali, questo passo determina il massimo numero di colori che i pixel possono avere. Ad esempio per un'immagine in bianco e nero (B/N) sono solitamente sufficienti 256 livelli di grigio, che si possono rappresentare in 8 bit. Il numero di bit necessari per rappresentare ciascun pixel è detto *bitrate*, si misura in bit/pixel (bpp) ed è un'importante metrica nella rappresentazione delle immagini digitali.

L'immagine così digitalizzata è ora pronta per la fase di *codifica* che consiste nella trasformazione in una rappresentazione più compatta del dato.

1.3.2 Spazio dei colori

Per memorizzare ed elaborare un'immagine a colori in un sistema digitale è necessario utilizzare un *modello di colore* ovvero un modello matematico che crea una corrispondenza biunivoca tra un colore e la sua rappresentazione numerica attraverso più canali cromatici (solitamente 3 o 4). Un'implementazione di un modello di colore è detta *spazio di colore* e definisce l'intervallo di colori, all'interno del modello, che possono essere rappresentati in quello spazio. In un modello di colore a 3 canali, utilizzando 256 livelli di colore per ciascun canale si può rappresentare il pixel in una gamma di più di 16 milioni di diversi colori (2^{24}) utilizzando 24 bit per pixel.

Uno dei modelli più noti è *RGB* (Red, Green and Blue ovvero rosso, verde e blu) introdotto dal CIE (Commission internationale de l'éclairage) negli anni '30. Essendo un modello a mescolanza additiva, si presta molto bene alla realizzazione dei display in quanto la luce emessa a diverse lunghezze d'onda si somma per generare una vasta gamma di colori. Per questo è ora

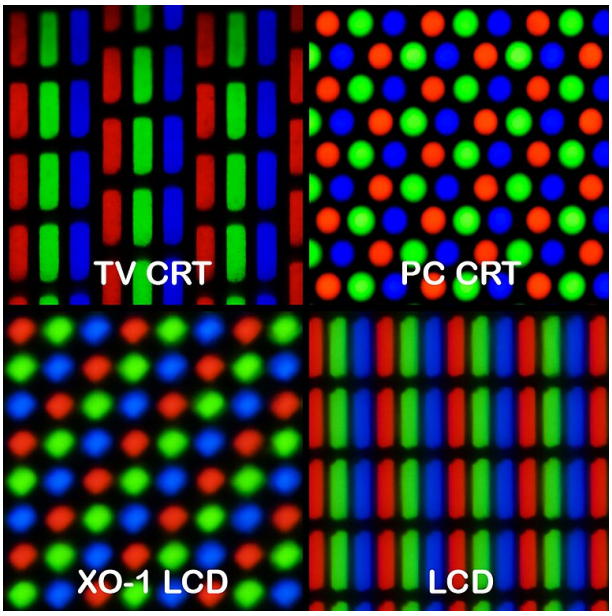


Figura 1.1: Schemi comuni di sub-pixel [7]

utilizzato nella quasi totalità dei monitor attraverso la giustapposizione di 3 microdisplay detti *sub-pixel* (uno per ogni canale) per formare un pixel (vedi fig. 1.1).

Un altro spazio di colore molto importante soprattutto nell'ambito della compressione dei dati è lo *YCbCr* (talvolta indicato con *YUV* che in realtà è la versione analogica utilizzata in ambito televisivo). YCbCr nasce come cambio di coordinate nello spazio RGB. I tre canali di YCbCr sono

- Y: luminanza o luminosità, è sostanzialmente la componente che permette di rappresentare l'immagine come scala di grigi
- Cb: che sta per *Chrominance blue* ovvero la differenza tra il canale B e quello Y
- Cr: che sta per *Chrominance red* ovvero la differenza tra il canale R e quello Y

Una caratteristica del sistema visivo dell'uomo è la maggiore sensibilità alla luminanza rispetto alle componenti di crominanza. È possibile sfruttare questo fatto per migliorare l'elaborazione e la compressione delle immagini digitali, ad esempio attraverso quello che è detto *chroma subsampling* o sottocampionamento della crominanza. Questa tecnica (utilizzata anche nella compressione JPEG) consiste nel mantenere le informazioni sulla crominanza ad una risoluzione inferiore rispetto a quelle sulla luminanza. È possibile effettuare questo sottocampionamento con diversi schemi, ciascuno dei quali implica un determinato livello di compressione. Ad esempio lo schema utilizzato da JPEG è il 4:2:0 che rappresenta tutti i campioni della luminanza ma solo un quarto di ciascun canale di crominanza. In questo modo si ha a tutti gli effetti una degradazione del segnale, che passa da un bitrate di 24 bpp a 12 bpp, ma questa distorsione

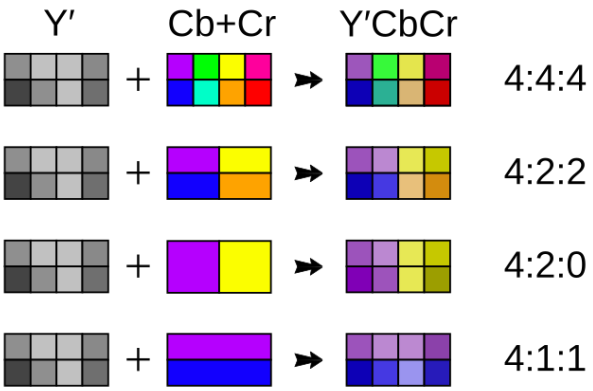


Figura 1.2: Chroma subsampling [8]

è praticamente invisibile all'occhio umano e per questo la rappresentazione YCbCr 4:2:0 è di gran lunga la più usata nel contesto della compressione video di contenuti "naturali". La stessa degradazione non è però accettabile per segnali di natura biomedicale o satellitare.

Capitolo 2

Metriche di valutazione di un sistema di compressione di immagini

Per poter comprendere e confrontare le tecniche attualmente in uso e quelle più avanzate è necessario studiare e conoscere le metriche che permettono di valutare vari aspetti di un sistema di compressione di immagini. Per questo motivo in questo capitolo vengono presentate in dettaglio le più note misure di valutazione che permettono di analizzare qualità e prestazioni del processo di compressione da vari punti di vista, come l'efficienza e la complessità computazionale della compressione, la fedeltà dell'immagine ricostruita e la percezione visiva.

Le metriche in questione posso generalmente essere suddivise in soggettive e oggettive. Per queste ultime si possono individuare due ulteriori categorie in base al fatto che misurino l'efficienza del sistema di compressione oppure la qualità del risultato.

2.1 Metriche oggettive per l'efficienza della compressione

2.1.1 Compression Ratio (CR)

Il *compression ratio* o "tasso di compressione" (indicato anche solamente con CR) è probabilmente la più semplice e immediata misura per un generico sistema di compressione (anche non specifico per le immagini). È un numero puro che indica quanto il sistema sia in grado di comprimere e si può quindi definire con la semplice formula

$$CR = \frac{\text{dimensione del dato non compresso}}{\text{dimensione del dato compresso}}$$

da cui è chiaro che si tratta di un valore positivo e maggiore di 1 se effettivamente avviene una compressione (ma in alcuni casi estremi può capitare che la compressione sia particolarmente

inefficiente, ad esempio su file molto piccoli, portando il dato ”compresso” ad essere più grande di quello originale, con la conseguenza che $CR < 1$).

Nella letteratura talvolta si utilizza una versione alternativa della stessa quantità, definita *compression factor* o ”fattore di compressione” (indicato anche con CF) che è semplicemente l'inverso del compression ratio ovvero

$$CF = \frac{\text{dimensione del dato compresso}}{\text{dimensione del dato non compresso}}$$

Un'ulteriore ed equivalente misura è il *tasso di codifica* che esprime, come un bitrate (bpp), il numero di bit per campione (pixel) o per secondo (nel caso di codifica di un video) del dato compresso. Ad esempio, un sistema che comprime un'immagine con un bitrate di 0.25 bpp utilizza solamente 0.25 bit per rappresentare un pixel (ovvero con un bit rappresenta 4 pixel). Chiaramente lo spazio di memoria occupato da un dato può essere calcolato anche come prodotto di bitrate e dimensione, ad esempio per un immagine bitrate per numero di pixel. Da questo deriva che il compression ratio e il compression factor possono essere anche calcolati come rapporti tra i bitrate prima e dopo la compressione.

Come anticipato nell'introduzione le tecniche lossy permettono di raggiungere tassi di compressione molto più elevati rispetto alle altre. Per le tecniche lossless, il compression ratio è in genere nell'ordine di grandezza di poche unità mentre per le tecniche con perdita si può arrivare tranquillamente a qualche decina e fino anche ad un centinaio. Questo estremo risparmio in termini di spazio rende, in alcuni casi, accettabile il compromesso con la qualità persa.

2.2 Metriche oggettive per la qualità della compressione

Parliamo ora delle metriche che quantificano la qualità oggettiva dell'immagine decompressa ovvero assegnano un numero alla fedeltà dell'artefatto all'immagine originale dopo il processo di compressione lossy.

2.2.1 MSE

Il *Mean Squared Error* o ”errore quadratico medio” (indicato anche con MSE) è un noto indicatore statistico commutativo che quantifica la distanza tra uno stimatore $\hat{\theta}$ e il dato stimato θ. Come suggerisce il nome, si calcola come media degli scarti quadratici ovvero in generale

$$MSE\left(\hat{\theta}\right) = E\left[\left(\hat{\theta} - \theta\right)^2\right]$$

Nel caso del confronto tra immagini, lo stimatore è l'immagine ricostruita dopo la compressione e decompressione mentre il dato stimato è l'immagine di partenza. Lo scarto quindi è la differenza pixel per pixel che c'è tra due immagini ovvero

$$MSE = \frac{\sum_{x=1}^{N} \sum_{y=1}^{M} [I(x,y) - I'(x,y)]^2}{M \times N}$$

per un'immagine B/N, dove

$$\begin{aligned} N &= \text{larghezza dell'immagine (in pixel)} \\ M &= \text{altezza dell'immagine (in pixel)} \\ I(x,y) &= \text{intensità del pixel dell'immagine originale alle coordinate } (x,y) \\ I'(x,y) &= \text{intensità del pixel dell'immagine ricostruita alle coordinate } (x,y) \end{aligned}$$

Per le immagini a colori si potrebbe pensare di utilizzare lo spazio di colore RGB e calcolare semplicemente la media di ogni colore

$$MSE_{RGB} = \frac{\sum_{x=1}^{N} \sum_{y=1}^{M} \sum_{c=1}^{3} [I(x,y,c) - I'(x,y,c)]^2}{M \times N \times 3}$$

dove la coordinata c indica il canale che si sta considerando. Una metrica più fedele alla qualità percepita dall'occhio umano però, è quella che si può ottenere passando prima allo spazio YCbCr, calcolando il MSE per ciascun canale (MSE_Y, MSE_{Cb}, MSE_{Cr}) e a questo punto combinare le tre metriche con una media aritmetica o ancor meglio con una media pesata in modo da dare più peso alla componente di luminanza. Un modo comunemente utilizzato per fare la media pesata è il seguente

$$MSE_{k11} = \frac{k \cdot MSE_Y + MSE_{Cb} + MSE_{Cr}}{k+2}$$

con $k = 6$ (come utilizzato nella parte sperimentale di questo elaborato) oppure $k = 4$ (come utilizzato nella libreria `libavfilter` di FFmpeg) [9].

In generale, se le immagini coincidono pixel per pixel si avrà MSE pari a 0 e più i pixel omologhi differiscono più il valore del MSE aumenta. È importante sottolineare però che la dissimilarità misurata dal MSE non sempre corrisponde a quanto percepito dall'occhio umano. Un esempio estremo è quello mostrato in figura 2.1 dove l'immagine di destra è stata creata appositamente come copia di quella a sinistra ma spostata a destra di un pixel. In questo modo ogni coppia di pixel omologhi avrà un certo errore e alla fine risulta MSE_{YCbCr}=155.63 sebbene ad occhio nudo, a parte la striscia nera di pixel sul lato sinistro, non si veda alcuna differenza. Il MSE infatti è un modello matematico molto semplice che non prende in considerazione

Figura 2.1: Immagini identiche per l'occhio umano ma con alto MSE

altri fenomeni che determinano la visione umana come la diversa sensibilità al contrasto, alla luminosità o al colore.

2.2.2 SNR e PSNR

Il *Signal to Noise Ratio* o "rapporto segnale-rumore" (SNR o S/N) è una grandezza fondamentale nella teoria dell'informazione e indica con un numero adimensionale (spesso espresso in scala logaritmica di decibel) la potenza del segnale utile rispetto a quella del rumore ovvero in generale

$$SNR = \frac{P_{signal}}{P_{noise}}$$

che per i segnali discreti diventa

$$SNR = 20 \log_{10} \frac{\sqrt{\frac{1}{n} \sum_{i=1}^{n} P_i^2}}{\sqrt{MSE}}$$

Questa misura può essere utilizzata anche per il confronto di immagini, dove la potenza indica l'intensità luminosa di ciascun pixel e MSE è la misura dell'errore definita in precedenza.

Per il confronto di immagini è particolarmente utilizzata una variante del SNR, detta *Peak Signal to Noise Ratio* (PSNR) e che si definisce come per il SNR ma considerando solo la massima intensità del segnale ovvero

$$PSNR = 20 \log_{10} \frac{I_{Max}}{\sqrt{MSE}}$$

dove I_{Max} può anche essere espressa come $2^b - 1$ con b numero di bit per esprimere il valore di un pixel. Possiamo quindi ricavare una formula semplificata che sarà per il caso di immagini

B/N con 8bpp

$$PSNR = 10 \log_{20} \frac{255^2}{\sqrt{MSE}} \approx 48dB - 10 \log_{10} MSE$$

Come detto per il MSE, nel caso di immagini a più canali si può procedere calcolando il PSNR per ciascun canale e infine combinare i risultati con una media aritmetica oppure, per un risultato più vicino all'esperienza visiva umana, con una media pesata in modo da dare più peso alle componenti maggiormente percepite (come la luminanza).

Il PSNR soffre delle stesse problematiche del MSE ma è comunque molto utilizzato per la sua semplicità di calcolo e di interpretazione. Si può infatti approssimare una relazione qualitativa tra la percezione dell'immagine compressa e il PSNR con quella originale: indicativamente, con un PSNR$\geq$45dB si ha una qualità dell'immagine ottima che rende sostanzialmente indistinguibili le due immagini, con un PSNR tra 40dB e 45dB la qualità è ancora molto buona e tra le due immagini ci sono differenze difficili da notare, con un PSNR tra 35dB e 40dB iniziano ad essere più marcate le differenze tra le due immagini pur mantenendo una qualità accettabile, con un PSNR tra 30dB e 35dB le differenze sono vistose e la qualità sufficiente mentre con un PSNR inferiore a 30dB la qualità è da considerarsi scarsa.

2.2.3 SSIM e MS-SSIM

Lo *Structural Similarity Index* o "indice di similarità strutturale" (SSIM) è una metrica che vuole quantificare la similarità tra due immagini in modo più fedele alla qualità soggettiva [10]. Per farlo utilizza i comuni indicatori statistici di media, varianza e covarianza di un blocchetto di $N \times N$ pixel tra l'immagine di riferimento e quella compressa. SSIM, a differenza di MSE e PSNR, cerca di rappresentare i fenomeni della percezione visiva come luminanza, contrasto e struttura che sono espresse nelle formule

$$l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_1}$$

$$s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

dove

$$\begin{aligned}
\mu_x, \mu_y &= \text{media rispettivamente di } x \text{ e di } y \\
\sigma_x^2, \sigma_y^2 &= \text{varianza di } x \text{ e di } y \\
\sigma_{xy} &= \text{covarianza di } x \text{ e } y \\
c_1 = (k_1 L)^2, c2 &= (k_2 L)^2, c_3 = c_2/2 \\
L &= \textit{dynamic range} \text{ dei valori dei pixel, usualmente pari a } 2^{\text{bpp}-1} \\
k_1 = 0.01, k_2 &= 0.03 \text{ valori di default}
\end{aligned}$$

Questi valori vengono poi combinati con dei pesi α, β, γ per ottenere

$$SSIM(x, y) = l(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(c, y)^\gamma$$

dove se $\alpha = \beta = \gamma = 1$ si ottiene

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

La variante *Multi-Scale SSIM* (MS-SSIM) consiste nel calcolare la metrica SSIM su più scale di risoluzione, ottenute sottocampionando per step successivi. [11]

2.2.4 Complessità computazionale

Un'altra metrica decisamente rilevante a lato pratico è la complessità computazionale della tecnica di compressione. A seconda del contesto di applicazione dell'algoritmo, può essere necessario che la compressione avvenga in un tempo critico particolarmente stringente (come succede in ogni smartphone ogni volta che viene scattata una fotografia) oppure il sistema potrebbe avere risorse hardware o energetiche limitate (si pensi ad un sistema posizionato in locazioni remote ed alimentato a batteria o a luce solare, ad esempio un satellite).

È possibile svolgere un'analisi teorica per conoscere la complessità di un algoritmo ma a lato pratico può essere più rilevante la misurazione dei tempi di codifica e decodifica. Questi sono infatti fortemente influenzati sia dall'implementazione dell'algoritmo che dalle ottimizzazioni hardware che possono essere messe in atto. Inoltre in un dispositivo multi thread come uno smartphone ci possono essere molte altre interferenze come processi nascosti che possono occupare tempo di calcolo all'algoritmo sotto esame.

È molto importante anche analizzare l'occupazione di memoria perché spesso quello che si vuole comprimere sono immagini di grandi dimensioni e un algoritmo non attento a questo aspetto potrebbe caricare tutto nella memoria principale. Sebbene i dispositivi moderni siano

ormai dotati di svariati GB di RAM, bisogna pensare anche a quei dispositivi equipaggiati con meno capacità di calcolo. Inoltre, caricare molti dati in memoria, supponendo che ci sia spazio, è comunque costoso in termini di tempo ed energia, ad esempio non sarebbe pratico e sostenibile per uno smartphone caricare centinaia di MB di parametri di una rete neurale solamente per comprimere un'immagine. Infine, riempire la memoria primaria porta necessariamente a dei page fault, che rallenterebbero di gran lunga le operazioni di codifica e decodifica dell'immagine.

Inevitabilmente, le tecniche più sofisticate di compressione richiedono più tempo e memoria per ottenere risultati migliori in termini di qualità e tasso di compressione. È quindi necessario scegliere un tradeoff tra compressione e capacità della tecnica sulla base della realtà di applicazione del sistema.

2.3 Metriche soggettive

L'obiettivo delle tecniche di elaborazione e compressione dei contenuti multimediali è sempre quello di fornire un servizio di alta qualità e non corrotto rispetto alla sorgente, sostanzialmente il più gradevole possibile per l'utente. Quando il fruitore del contenuto è un umano, la metrica più precisa per misurare la qualità è proprio quella che consiste nel chiedere all'umano un'opinione soggettiva sul contenuto. Tutte le tecniche e i modelli matematici mostrati finora cercano di automatizzare questo processo in quanto le metriche soggettive sono da un lato le più rilevanti in molti casi, dall'altro sicuramente le più costose in termini di tempo e denaro.

Per ottenere un risultato statisticamente rilevante è necessario chiedere l'opinione di un ricco numero utenti, il che può richiedere molto tempo. È inoltre essenziale garantire le medesime condizioni di fruizione del contenuto per tutti gli utenti che formulano un giudizio ad esempio, nel caso della valutazione delle immagini, è opportuno eseguire il test con lo stesso schermo, distanza, illuminazione e metodologia. Non è da sottovalutare nemmeno l'aspetto psicologico in quanto è ben possibile che più utenti siano indotti a dare un giudizio alterato ad una particolare immagine ad esempio se posta in prossimità spazio-temporale di altre particolari immagini, oppure se lo stimolo non è di loro gradimento o suscita in essi una risposta emotiva.

È possibile condurre il test mostrando agli utenti un'immagine per volta oppure più immagini a confronto. Un metodo per far valutare all'utente una serie di immagini è il cosiddetto *Absolute Category Rating* (ACR) che consiste nel mostrare i campioni compressi senza il riferimento dell'immagine originale. In alternativa si può decidere di inserire all'interno del test anche l'immagine non compressa, senza però indicare all'utente quale sia. In questo caso la tecnica è detta ACR-HR ovvero *Absolute Category Rating with Hidden Reference* e permette di valutare la coerenza e l'affidabilità delle valutazioni espresse. Una scelta comune nell'applicazione di ACR e ACR-HR è quella di fornire all'utente 5 livelli di gradimento, ciascuno associato al suo

valore numerico: ottimo, buono, accettabile, mediocre e scarso. Con queste misure si può poi calcolare il *Mean Opinion Score* o ”punteggio medio d’opinione” (MOS) come semplice media aritmetica dei punteggi dati dai singoli utenti.

Le sessioni di valutazione spesso iniziano con una fase di training dove l’utente familiarizza con la strumentazione ed il setup, segue poi la cosiddetta *sequenza di stabilizzazione* ovvero un piccolo numero di dati che non confluiranno nel rating del servizio ma servono per stabilizzare l’opinione dell’utente e solo a questo punto inizia la vera e propria raccolta di opinioni.

Capitolo 3

Stato dell'arte dei metodi di compressione tradizionali

Prima di vedere come operano le tecniche di compressione che usano le reti neurali, è utile capire funzionamento, performance e limiti delle più comuni codifiche "tradizionali" per immagini.

3.1 JPEG

Il *Joint Photographic Experts Group* (JPEG) [13] è il più comune e diffuso formato di memorizzazione e compressione di immagini. Permette di comprimere immagini sia in modalità lossy che lossless ma quest'ultima è molto meno conosciuta e utilizzata in quanto per la codifica senza perdita sono molto più diffusi i formati come PNG, TIFF o BMP. JPEG prende il nome dal comitato di esperti ISO e CCITT che ne ha sviluppato e pubblicato le specifiche nel 1992. JPEG è uno standard ISO (ISO/IEC 10918-1) che permette l'interoperabilità tra implementazioni diverse. JPEG permette di comprimere immagini B/N oppure a colori ma non con la componente di trasparenza o "alpha channel" (come invece fa PNG).

Figura 3.1: Logo JPEG [12]

3.1.1 Il processo di codifica

Sebbene non sia previsto dallo standard, un'operazione di preprocessing, che viene spesso seguita dalle implementazioni dei codificatori JPEG, è il chroma subsampling (vedi 1.3.2) in quanto permette una compressione significativa senza una grossa perdita di qualità per l'occhio umano. La specifica vera e propria non distingue i canali quindi da qui in avanti è indifferente lo spazio di colori e si possono trattare separatamente allo stesso modo tutte le componenti.

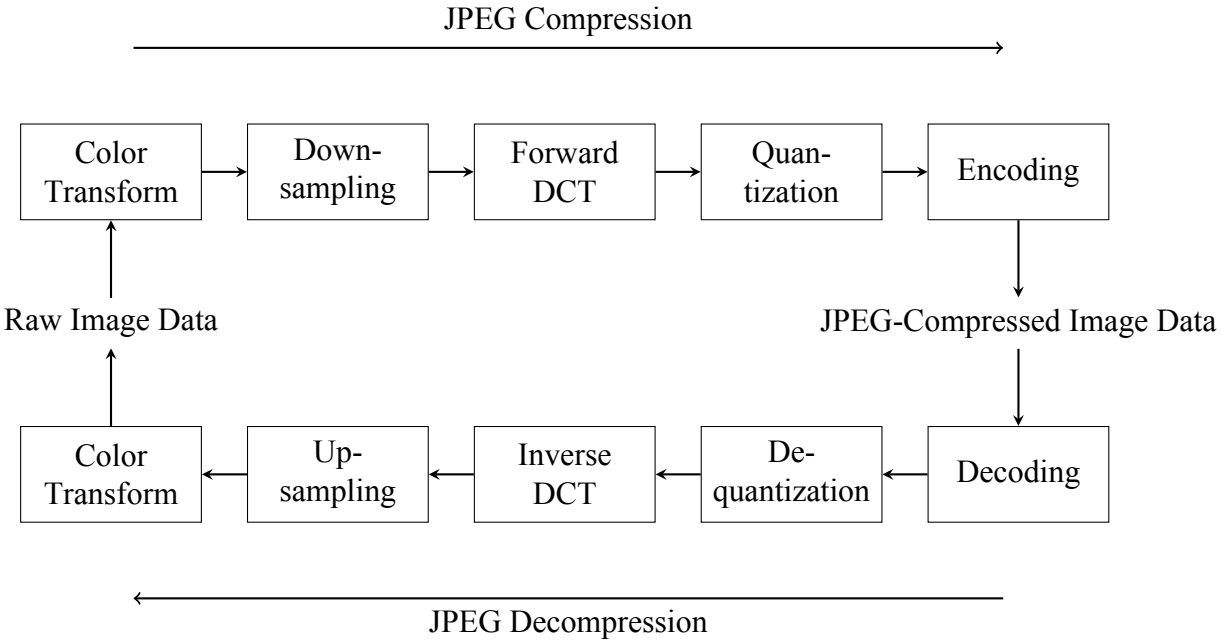


Figura 3.2: Diagramma di funzionamento di JPEG

Il processo di codifica JPEG inizia con la suddivisione dell'immagine in blocchi di 8×8 pixel. Su ciascun blocco vengono poi effettuate tre operazioni: DCT, quantizzazione e codifica lossless.

La *Discrete Cosine Transform* o "trasformata coseno discreta" (DCT) è una trasformata lineare e ortogonale proposta nel 1972 da Ahmed, Natarajan e Rao [14]. Può essere considerata una variante della DFT (Discrete Fourier Transform) infatti gli autori stessi hanno proposto un algoritmo per calcolarla che si basa sulla FFT (Fast Fourier Transform). A differenza della DFT, la DCT produce solo valori reali infatti il k-esimo coefficiente $G_x(k)$ della DCT di una sequenza di dati $X(m)$ con $m = 0,1, ..., (M-1)$ è definito nel seguente modo

$$G_x(0) = \frac{\sqrt{2}}{M} \sum_{m=0}^{M-1} X(m)$$

$$G_x(k) = \frac{2}{M} \sum_{m=0}^{M-1} X(m) \cos \frac{(2m+1)k\pi}{2M} \qquad k = 1,2, ..., (M-1)$$

La caratteristica più importante della DCT che ne ha reso significativo l'utilizzo nelle tecniche di compressione è il fatto che sia una trasformata **sparsificante** ovvero in grado di trasformare un segnale con supporto in frequenza concentrato in un segnale sparso cioè composto da pochi campioni non trascurabili e molti trascurabili. Questa proprietà dei segnali è molto utile nella compressione in quanto permette di individuare e mantenere solo le poche informazioni più significative (o di memorizzare con un diverso livello di dettaglio in base all'importanza del-

l'informazione). La scelta di usare blocchi 8×8 permette alla DCT di cogliere le caratteristiche locali dell'immagine ma non le relazioni che ci possono essere tra zone più lontane di pixel.

Lo step successivo nella codifica JPEG è la quantizzazione, l'unica componente che determina la natura lossy del sistema di compressione. Lo standard permette di utilizzare una tabella di quantizzazione $q(i, j)$ che definisce per ogni frequenza spaziale il passo di quantizzazione Δ. Sostanzialmente la quantizzazione consiste nella divisione e arrotondamento della matrice ottenuta dalla DCT per il valore corrispondente della tabella. Anche in questo caso è possibile sfruttare alcune caratteristiche del sistema della visione umana, in particolare la maggiore sensibilità alle basse frequenze spaziali ovvero le variazioni graduali su grandi pezzi di immagine. Scegliendo passi di quantizzazione più piccoli per queste frequenze, è possibile rappresentarle in maggior dettaglio raggiungendo quindi un risultato finale più nitido. Allo stesso modo è possibile usare passi di quantizzazione più grandi per le alte frequenze, consentendo di risparmiare molti bit per pixel. È possibile inoltre definire tabelle diverse a seconda del canale che si sta considerando ad esempio, pur non imponendo i valori delle tabelle di quantizzazione che sono una libera scelta dell'implementatore, lo standard suggerisce la seguente tabella per la luminanza a seguito di test soggettivi

$$q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 81 & 61 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 111 & 90 \\ 49 & 63 & 78 & 87 & 101 & 121 & 120 & 100 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Da notare che i valori più piccoli dei passi di quantizzazione sono nell'angolo in alto a sinistra, dove sono rappresentate le basse frequenze. Le tabelle, non essendo specificate dallo standard, vanno sempre incluse nel file codificato. Possono poi essere scalate per ottenere diversi livelli di qualità (e di compressione) dell'immagine.

L'ultimo step della compressione JPEG è la codifica lossless. In questo passaggio si vogliono tramutare i valori quantizzati della DCT in una stringa di bit che possa essere memorizzata su di un file. Questa fase può essere ulteriormente scomposta in due passaggi: la linearizzazione dei valori e la codifica vera e propria. Il risultato della quantizzazione di un blocco 8×8 è a sua volta un blocco 8×8 e per essere memorizzato efficientemente è necessario trasformarlo in una stringa di valori. Per farlo si può pensare di scorrere la matrice in un qualsiasi modo (row-major, column-major etc.) ma per la natura della DCT la cosa più efficiente è il cosiddetto *zig-zag scan*

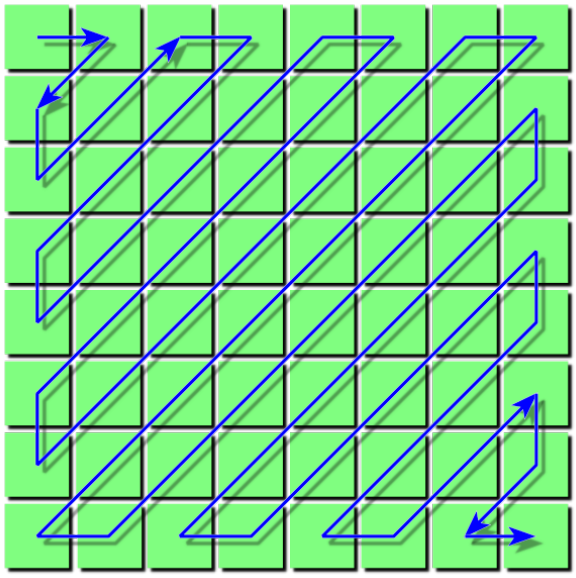


Figura 3.3: Zig-Zag scan [15]

(vedi 3.3) in quanto la sparsificazione seguita dalla quantizzazione rendono nulli molti dei valori che non sono nell'angolo in alto a sinistra. Si ottiene in questo modo una sequenza che in generale termina con molti zeri, questo si può rappresentare indicando con un simbolo apposito EOB (*End Of Block)* la fine dei valori non nulli. A questo punto risulta particolarmente conveniente usare lo *Zero Run Lenght Encoding* (ZRLE), una variante del *Run Lenght Encoding* (RLE) specializzata per le sequenze ricche di valori nulli, che consiste nel rappresentare la sequenza di valori come coppie (N, V) dove N è il numero di zeri presenti nella sequenza prima del valore V. Ad esempio se la sequenza linearizzata dallo zig-zag scan è $(0,0,1,1,0,2, EOB)$, può essere rappresentata secondo ZRLE come $(2,1), (0,1), (1,2), (0,0)$ [1]. A questo punto si può codificare la sequenza ottenuta tramite ZRLE, ad esempio[2] codificando prima tutti i valori tramite la rappresentazione categoria-valore e infine le coppie tramite una codifica entropia a prefisso.

Come si può vedere nella figura 3.2, la decodifica esegue al contrario tutti i passaggi della codifica. In particolare DCT e codifica lossless sono invertibili mentre la quantizzazione no. Il processo di dequantizzazione consiste nel moltiplicare ciascun blocco decodificato per la tabella di quantizzazione, ottenendo così le approssimazioni dei valori originali.

[1] $(0,0)$ indica il simbolo EOB

[2] Anche in questo caso lo standard non specifica la codifica quindi vi è libera implementazione ma nel file compresso vanno inserite le informazioni sufficienti per eseguire la decodifica

(a) Immagine originale [16]

(b) Immagine compressa [17]

Figura 3.4: Effetto della compressione aggressiva di JPEG con generazione di artefatti

3.1.2 Pro e contro di JPEG

La codifica JPEG ha dalla sua parte la grande diffusione che ha raggiunto negli anni con conseguente supporto e compatibilità pressoché universale. Nonostante le sue limitazioni, è difficile che una tecnica così comune e utilizzata possa essere sostituita da un'altra, anche se più performante.

JPEG infatti è una codifica molto semplice e veloce dal punto di vista computazionale, anche perché può sfruttare le implementazioni ultra ottimizzate della FFT per calcolare la DCT. La qualità visiva di immagini fotografiche si può considerare buona o molto buona anche con dimensioni relativamente ridotte, ad esempio scegliendo un compression ratio $\approx$10, mentre con tassi più alti la qualità decade e si possono venire a generare degli artefatti di compressione ovvero alterazioni dell'immagine particolarmente lampanti e sgradevoli all'occhio umano (vedi 3.4).

3.2 JPEG 2000

JPEG2000 (indicato anche con JP2) è uno standard (ISO/IEC 15444-1:2019) per la compressione di immagini nato tra il 1997 e il 2000 dal comitato JPEG come evoluzione del precedente standard JPEG del 1992 e pubblicato nel 2002 [19]. Come anche JPEG, JPEG2000 è in grado di produrre una compressione sia lossy che lossless.

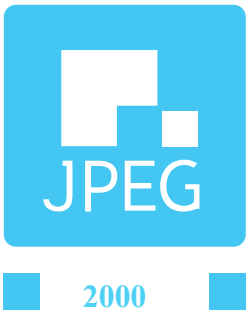


Figura 3.5: Logo JPEG2000 [18]

3.2.1 Differenze con JPEG

JPEG2000 usa la DWT (*Discrete Wavelet Transform*) ovvero una trasformata invertibile che, a differenza della DFT su cui si basa anche la DCT utilizzata da JPEG, rappresenta il segnale come somma di wavelets ovvero funzioni oscillatorie localizzate che possono essere scalate e spostate nel tempo così da portare sia informazioni

temporali che di frequenza in un modo più flessibile rispetto alle altre due trasformate. Grazie a questo è in grado di preservare più dettagli e ridurre gli artefatti anche ad alti tassi di compressione.

Mentre JPEG divide e codifica sempre blocchi 8×8, JPEG2000 permette la presenza di *tiles*[3] rettangolari codificate indipendentemente e con la possibilità di avere diversi livelli di compressione per ciascuna di esse. L'indipendenza delle tiles permette di codificare l'immagine sfruttando la parallelizzazione delle architetture. In generale, tiles più grandi permettono una maggiore compressione e migliore rendimento dal punto di vista degli artefatti ma richiedono più memoria per l'elaborazione quindi impostando la dimensione è possibile scegliere il tradeoff tra compression ratio e complessità computazionale.

JPEG2000 è anche in grado di comprimere le immagini a diversi livelli di risoluzione, oltre che di qualità. Sia risoluzioni che qualità diverse possono essere codificate contemporaneamente per la stessa immagine tramite le tecniche dette di *resolution scalability* e *quality scalability*. Questo permette di supportare la trasmissione progressiva, una tecnologia che consiste nella possibilità di visualizzare una prima versione a bassa risoluzione dell'immagine prima ancora che tutto il file sia stato trasmesso e, mano a mano che arrivano ulteriori dati, consente di migliorare il file decodificato. In particolare è anche possibile mettere in atto una trasmissione progressiva da lossy a lossless, ovvero codificare all'interno di uno stesso data stream inizialmente una versione lossy per poi arrivare, a mano a mano che arrivano più dati, alla versione lossles. Queste tecniche permettono a JPEG2000 di essere più adeguato per le applicazioni scalabili.

JPEG2000 inoltre permette di lavorare con immagini ad altissima risoluzione senza separarle in blocchi totalmente indipendenti. Questo è fatto per evitare il più possibile gli artefatti che invece si vengono a creare nella codifica JPEG ed è possibile grazie alle caratteristiche della DWT multi scala e alle diverse dimensioni delle tiles.

Anche la fase di codifica è migliorata rispetto a quella di JPEG. Per JPEG2000 si ha una codifica su due livelli: prima una codifica ad entropia che utilizza un metodo detto *Embedded Block Coding with Optimal Truncation* (EBCOT) e successivamente la stringa di bit è ulteriormente compressa organizzando i dati in pacchetti ottimizzando così la scalabilità e la resilienza agli errori per favorire una trasmissione più robusta e una migliore gestione delle risorse.

A differenza di JPEG, JPEG2000 supporta nativamente la trasparenza attraverso l'aggiunta di metadati. Questo è particolarmente importante per chi lavora con sistemi di grafica su più layer.

[3] letteralmente "piastrelle"

3.2.2 Pro e contro di JPEG2000

Nonostante la superiorità di JPEG2000 rispetto alla precedente versione evidenziate nel paragrafo precedente, la sua diffusione resta piuttosto limitata a causa della predominanza di quest'ultima. Il supporto a JPEG2000 infatti è ancora limitato o assente, ad esempio nei browser. È comunque utilizzato in alcuni ambiti dove è importante la compressione lossless oppure lossy ad alta qualità come nelle immagini a scopo medico, di videosorveglianza, del cinema digitale o del mercato della trasmissione video [20].

Capitolo 4

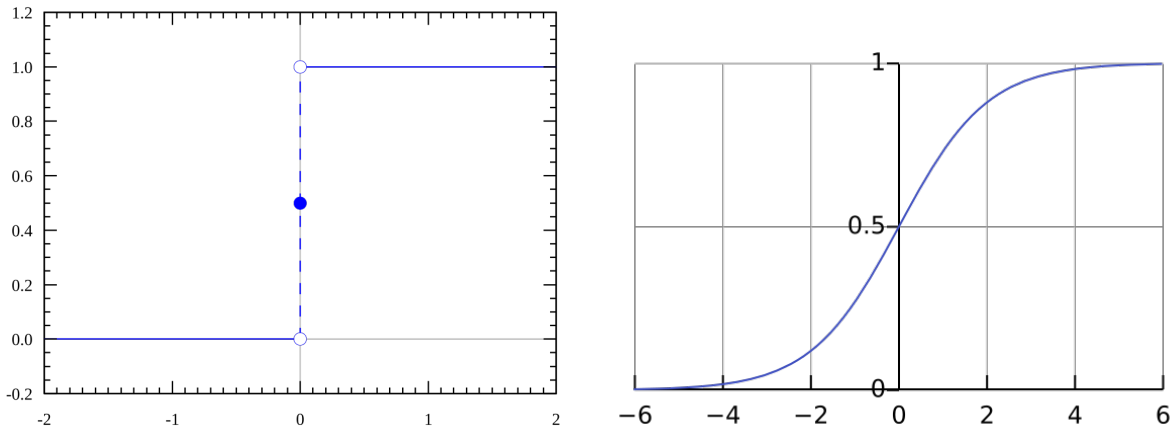
Reti neurali

4.1 Introduzione

Le *Neural Network* o *Artificial Neural Network* (NN o ANN) sono dei modelli matematico-informatici, frutto del lavoro interdisciplinare tra matematica e neuroscienze, ispirati alla struttura biologica dei cervelli animali. Il concetto biologico di neurone è mappato nel cosiddetto *neurone artificiale*, un nodo della rete connesso agli altri attraverso dei rami che simulano il funzionamento di dendriti e assone.

Nella storia delle reti neurali possiamo individuare tre momenti chiave. Il primo è quando, nel 1943, viene proposto il primo modello di rete neurale da Warren S. McCulloch e Walter Pitts [21] avanzando l'ipotesi secondo la quale sia possibile modellare l'attività dei neuroni del cervello come funzioni binarie che si attivano solo dopo una certa soglia, suggerendo quindi una similarità con il funzionamento dei computer che stavano nascendo proprio in quegli anni. Poi nel 1957 Frank Rosenblatt inventa il concetto di **percettrone** come unità della rete neurale che funge il ruolo di classificatore binario [22]. Sostanzialmente dati N valori in input, il percettrone ne computa la media pesata e poi la passa ad una funzione di attivazione, ovvero una mappatura nell'intervallo $[0,1]$. L'uscita di un neurone si può quindi calcolare come

$$h_i = \sigma\left(\sum_{j=1}^{N} w_{ij}x_j + c_i\right)$$



(a) Funzione gradino [23]

(b) Sigmoide logistica [24]

Figura 4.1: Esempi di funzioni di attivazione

dove

x_j j-esimo input del neurone

w_{ij} peso del ramo i-esimo entrante nel neurone j-esimo

c_i bias weight

σ funzione di attivazione

Le funzioni di attivazione possono essere di molti tipi: la più semplice è il gradino (vedi 4.1a) che implementa perfettamente il concetto di classificatore binario. Un'altra funzione comunemente usata è la sigmoide logistica (vedi 4.1b) che ha equazione

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

ed è particolarmente importante in quanto continua e derivabile.

Il terzo momento rilevante nella storia delle reti neurali è l'associazione delle reti di percettroni con il concetto di *machine learning* o "apprendimento automatico" [25] ovvero la branca dell'intelligenza artificiale che utilizza grandi moli di dati e le loro correlazioni statistiche per "imparare". Grazie a queste tecniche, i pesi che regolano il funzionamento di ogni percettrone possono essere imparati automaticamente dalla macchina attraverso gli algoritmi di backpropagation. In questo modo la rete neurale passa dall'essere una complessa funzione matematica resa non lineare dalle funzioni di attivazione, all'essere un sistema "intelligente" che ha lo scopo di dare le migliori risposte possibili anche per valori di input sconosciuti.

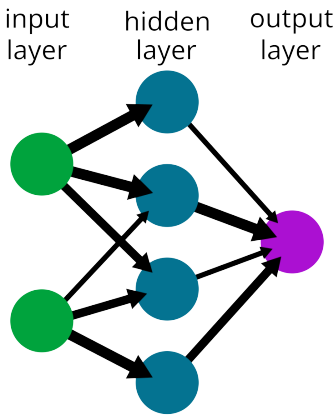


Figura 4.2: Semplice rete neurale [26]

4.2 Classi di reti neurali

4.2.1 MLP

La più semplice rete neurale possibile è composta da un layer di input ovvero una rappresentazione dei dati di ingresso, un layer nascosto ovvero una serie di percettroni collegati al layer di input tramite dei rami e infine un layer di output che raccoglie le uscite dei perceptron ricavando il risultato della rete (vedi 4.2). Ogni ramo che collega un nodo del layer di input ad un percettrone ha un peso associato ad esso che serve al percettrone stesso per calcolare la media ponderata. Secondo il teorema di approssimazione universale, una rete così composta (un solo hidden layer) è in grado di approssimare qualsiasi funzione continua in un intervallo chiuso e limitato, a patto che ci sia un numero sufficiente di neuroni. Anche con una rete così semplice è quindi possibile risolvere problemi non linearmente separabili (cosa che invece non è possibile per un singolo percettrone).

Le *Multi Layer Perceptron* (MLP) nascono come estensione della semplice rete neurale discussa al paragrafo precedente e come dice il nome stesso sono composte da più di un hidden layer di percettroni (vedi 4.3). Questo permette di approssimare funzioni complesse con un numero minore di nodi e connessioni rendendo la computazione più efficiente. Inoltre la separazione in layer permette una rappresentazione dell'input su più livelli di astrazione consentendo quindi di cogliere più a fondo la natura del dato di ingresso e riuscendo così a ricavarne risultati più complessi. Le MLP sono reti di tipo *feedfoward* ovvero reti nelle quali il flusso di dati prosegue linearmente dal layer di input a quello di output, senza cicli o connessioni di feedback, a differenza di quello che succede nelle RNN.

Uno tra i primi esempi di utilizzo di una MLP come sistema di compressione per immagini è quello proposto nel 1989 da Sonehara *et al.* [28]. Il modello non si discosta molto da quelli che sono i macro step tradizionali di un sistema di compressione: trasformazione in uno spazio

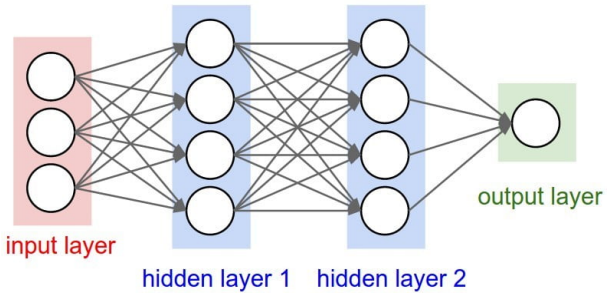


Figura 4.3: Multi Layer Perceptron [27]

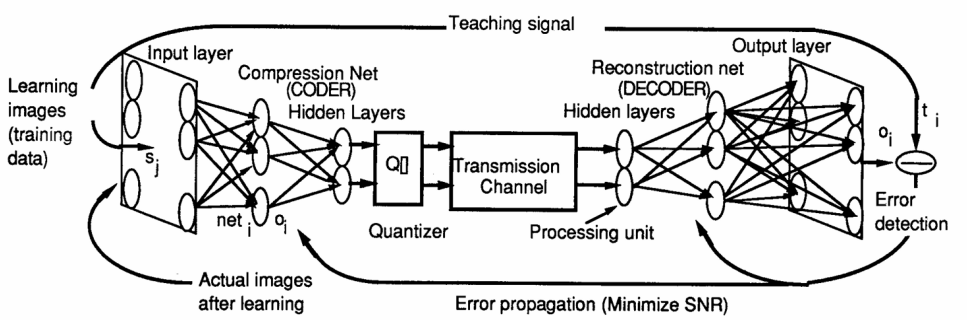


Figura 4.4: Modello proposto da [28]

latente, quantizzazione e codifica. L'implementazione pone però l'attenzione sulla riduzione di dimensionalità dell'immagine che viene ottenuta grazie ad una prima MLP feedforward con tre layers (vedi 4.4) che confluiscono in un layer ridotto ovvero con meno nodi rispetto a quello di input. Similmente nella fase di decodifica si ha una rete analoga che ha il compito di ricostruire l'immagine a partire dalla rappresentazione latente. La rete è stata allenata tramite gradient descent per minimizzare una versione scalata del MSE.

Questa struttura a "bottleneck" può essere considerata l'implementazione di un *Auto Encoder* (AE) dove la rappresentazione latente nello strato ridotto costituisce la versione compressa dell'immagine.

4.2.2 CNN

Le *Convolutional Neural Network* (CNN) o "reti neurali convoluzionali" sono un tipo di rete neurale che prende il nome dall'operazione di convoluzione (solitamente bidimensionale) che avviene all'interno dei suoi hidden layers. Le CNN si basano sui cosiddetti *kernel* o filtri ovvero piccole matrici di valori che vengono fatte scorrere sulla matrice di input (vedi 4.5). Ogni valore della matrice di uscita viene calcolato come combinazione lineare dei dati (pixel) ad esso adiacenti in quella di ingresso, attraverso i pesi determinati dal kernel. Questa operazione consente di individuare delle features all'interno dell'input. I kernel possono essere composti

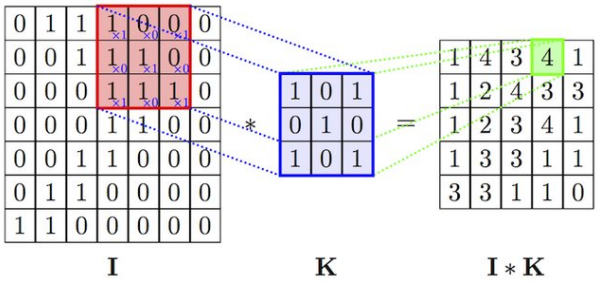


Figura 4.5: Esempio di convoluzione 2D [29]

da valori predefiniti per riconoscere particolari features note oppure possono essere inizializzati randomicamente e poi aggiustati tramite il processo di apprendimento automatico.

Le features che la CNN è in grado di riconoscere sono sempre più complesse mano a mano che il flusso di dati passa nei layer più profondi. Ad esempio al primo livello è possibile riconoscere i bordi di un oggetto e poi tramite la combinazione di questi possono essere individuate la texture oppure semplici forme geometriche. Nei layer più profondi, dove il *receptive field* è più grande, si riescono ad identificare parti di oggetti o volti oppure la loro totalità.

Le CNN sono largamente utilizzate nell'ambito della *Computer Vision*, in particolare per risolvere task di rilevamento di oggetti, segmentazione e classificazione di immagini, riconoscimento dei volti o delle azioni all'interno di video. L'abilità delle CNN di rispondere a questo tipo di bisogni le rende comunemente impiegate anche all'interno di sistemi per l'analisi di immagini a scopo medico, per il rilevamento di anomalie o nei sistemi di guida autonoma.

Anche nell'ambito della compressione di immagini ci sono alcuni studi che utilizzano le CNN come quello proposto nel 2016 da Ballé *et al.* [30]. Il sistema consiste in una trasformazione di analisi non lineare, un quantizzatore uniforme e una trasformazione di sintesi non lineare. Le trasformazioni sono formate da tre stage, ciascuno composto da filtri convoluzionali, sottocampionamento e normalizzazione divisiva con l'obiettivo di ottenere una rappresentazione dell'input in uno spazio con dimensionalità ridotta. Le CNN sono infatti in grado di codificare l'immagine in uno spazio latente grazie alla capacità dei layer convoluzionali di catturare le dipendenze spaziali e di estrarre le features dell'immagine. La particolarità del metodo proposto consiste nell'ottimizzazione *end-to-end* ovvero l'allenamento congiunto di tutte le fasi del processo, compressione, quantizzazione, codifica e decompressione che avviene tramite stochastic descent con l'obiettivo di minimizzare una loss function composta da una media pesata di tasso di codifica e distorsione. I risultati mostrano un migliore tradeoff tra tasso e distorsione rispetto alle codifiche tradizionali come JPEG e JPEG2000 e anche rispetto ad altri approcci con reti neurali sia secondo le metriche di MSE (PSNR) che MS-SSIM come mostrato in figura 4.6. Gli studi hanno trovato che il guadagno percentuale persiste per tutte le immagini testate ed è notevole che, nonostante la metrica di distorsione usata per l'ottimizzazione sia l'MSE, le immagini

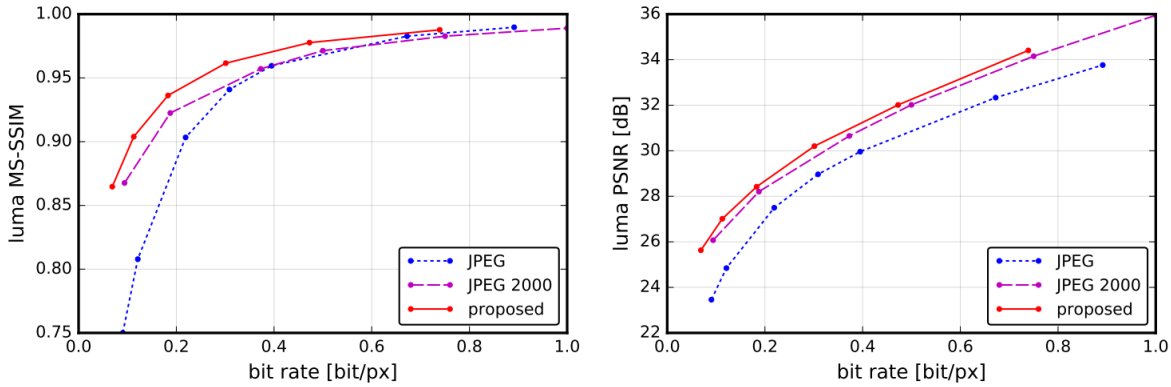


Figura 4.6: Tradeoff bitrate-distorsione con MS-SSIM e luma PSNR in [30]

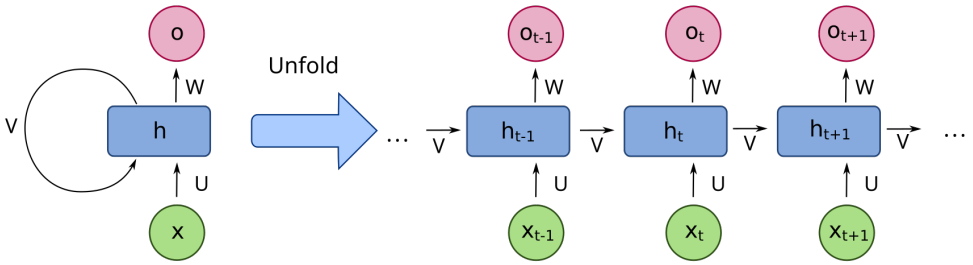


Figura 4.7: Schema di una RNN [31]

ottenute risultino più naturali di quelle compresse con JPEG e JPEG2000.

4.2.3 RNN

Le *Recurrent Neural Network* (RNN) sono reti che permettono connessioni cicliche al loro interno. Questa tecnica permette di mantenere qualche forma di memoria o di stato all'interno della rete detto *hidden state*. Di fatto instaurano delle connessioni tra i neuroni attraverso il tempo e per questo sono utilizzate principalmente nell'elaborazione di dati sequenziali. Le RNN possono essere rappresentate evidenziando la ciclicità del flow dei dati oppure "srotolando" nel tempo gli stati della rete (vedi fig 4.7). Una semplice formulazione matematica per descrivere le RNN è la seguente

$$h_t = \sigma\left(W_h h_{t-1} + W_x x_t + b_h\right)$$
$$y_t = W_y h_t + b_y$$

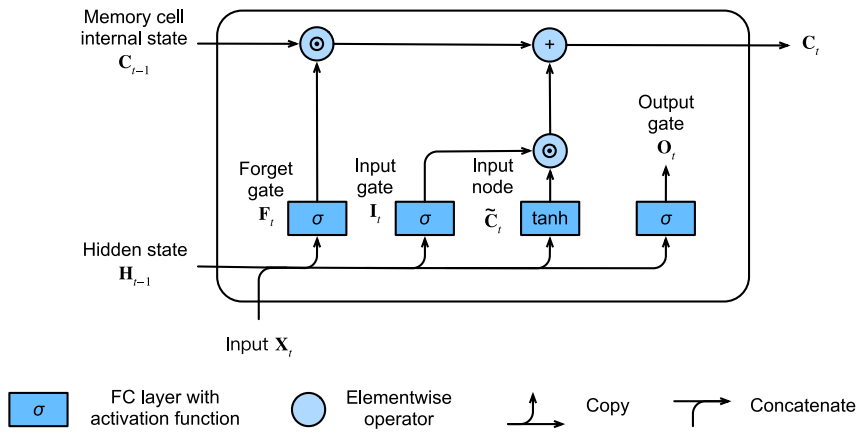


Figura 4.9: Cella di memoria di una LSTM [33]

dove

x_t è l'input della rete al tempo t

h_t è l'hidden state al tempo t

W_h, W_x, W_y sono le matrici dei pesi

b_h, b_y sono i bias

σ è la funzione di attivazione

L'allenamento di una rete ricorrente può essere condotto con le tecniche classiche di back-propagation ma con alcune accortezze note come tecniche di *Backpropagation Through Time* (BPTT). Innanzitutto è necessaria la rappresentazione srotolata della rete in modo da poter calcolare il gradiente ad ogni istante temporale per aggiornare i pesi. Inoltre avendo virtualmente una profondità molto elevata nel tempo, si possono manifestare i due problemi opposti di *vanishing gradients* oppure *exploding gradients*. Per risolvere questi problemi è stata inventata la *Long Short-Term Memory* (LSTM) [32] che utilizza tre funzioni dette *input gate*, *forget gate* e *output gate* per controllare il flusso di dati nella cella di memoria come mostrato in figura 4.9.

Un importante lavoro sulla compressione con le RNN è quello proposto da Toderici *et al.* nel 2017 [34]. Le architetture proposte si compongono di tre fasi: codifica, binarizzazione stateless e decodifica. Il funzionamento della tecnica in esame sfrutta la memoria delle RNN per mantenere l'errore residuo. Sostanzialmente per codificare un'immagine la si passa attraverso codificatore e binarizzatore per ottenere una prima approssimazione, poi questa viene decodificata e si calcola l'errore rispetto al dato originale. L'errore viene reinserito come input e segue la stessa procedura dell'immagine originale, così facendo si ottengono stream di bit che permettono di avere una qualità sempre più alta, in modo non molto diverso dalla codifica progressiva messa in atto da JPEG e JPEG2000. Una volta ottenuto lo stream completo si procede a ritro-

so ovvero decodificando una prima versione a bassa qualità dell'immagine e poi mano a mano decodificando gli errori per migliorare il risultato. I ricercatori hanno trovato valori più alti di MS-SSIM e PSNR con le loro architetture rispetto alle codifiche tradizionali, sia effettuando il training su un dataset di immagini 32×32 che su un dataset denominato *High entropy* ottenuto dai campioni 32×32 ricavati dalle immagini di un altro dataset 1280×720 che ottengono il peggior compression ratio quando codificati con PNG.

4.2.4 GAN

Le *Generative Adversarial Network* (GAN) sono delle reti neurali composte da due entità, un generatore e un discriminatore, che si allenano l'un l'altro. Il generatore produce dei dati con l'obiettivo di renderli il più possibile simili ai dati reali in modo da "ingannare" il discriminatore che invece tenta sempre di distinguere i dati reali da quelli generati artificialmente. Le due reti sono solitamente allenate separatamente e alternativamente. Ad esempio si può fissare il generatore e allenare un primo discriminatore capace di distinguere un rumore randomico dai segnali reali per poi fissare questo discriminatore ed allenare il generatore, procedendo in questo modo per più iterazioni.

Le GAN sono uno tra i più promettenti strumenti dell'intelligenza artificiale. In particolare stanno dimostrando ottimi risultati nei campi della generazione di immagini e video, data augmentation, style transfer e unsupervised learning. Nonostante ciò, pongono diverse sfide. Possono infatti venirsi a creare difficoltà durante la fase di allenamento in quanto vi è il rischio che generatore e discriminatore non convergano ad una prestazione ottimale ma continuino a sovrastarsi l'un l'altro, ad esempio se il generatore diventa troppo abile allora il discriminatore non è più in grado di distinguere il vero dal falso e per causa del vanishing gradient non si riesce ad uscire da quella situazione. Possono infine verificarsi i cosiddetti *mode collapse*, situazioni nelle quali la rete si stabilizza su pochi punti di lavoro ottimale rendendo scarse le prestazioni al di fuori di questi local optima e facendo quindi oscillare il sistema tra poche modalità di funzionamento.

La metrica utilizzata per testare il modello è MS-SSIM e per avvicinarsi ancora di più alla percezione umana è stato utilizzato lo spazio di colori YCbCr con sottocampionamento 611. I risultati mostrano valori di MS-SSIM più alti rispetto a tutte le altre metriche considerate, pur mantenendo un tempo di esecuzione volutamente basso per lasciare la possibilità di diffusione del sistema in applicazioni comuni.

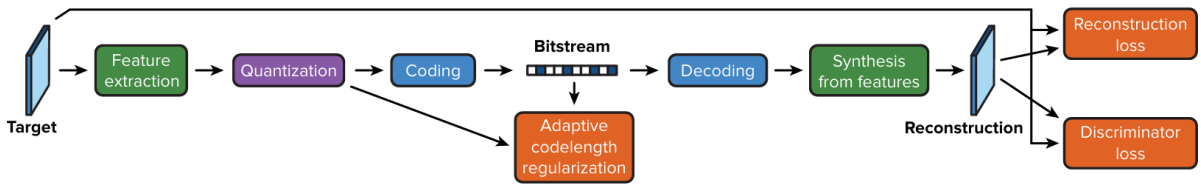


Figura 4.10: Struttura del modello presentato da [35]

Capitolo 5

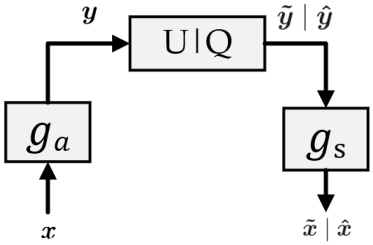
Esempio pratico di compressione: implementazione e risultati

Si mostra ora un lavoro più recente per la compressione di immagini con l'utilizzo di reti neurali.

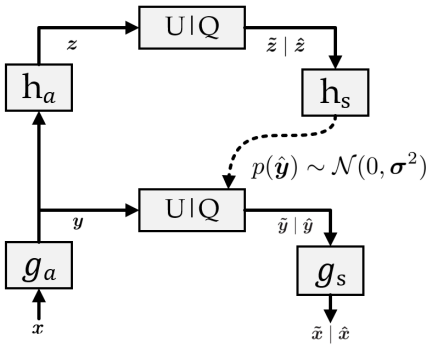
5.1 Concetto e funzionamento del metodo con DGML e Attention modules

Cheng *et Al.* [38] hanno proposto nel 2020 un'architettura innovativa che vuole migliorare l'accuratezza dei modelli di probabilità per la codifica entropica, fondamentale per una compressione efficiente delle immagini. Il modello proposto nasce come evoluzione di alcuni lavori precedenti (vedi figura 5.1). L'innovazione principale è l'uso di un modello *Gaussian Mixture Likelihoods* discretizzato (DGML) per stimare e parametrizzare la distribuzione di probabilità delle rappresentazioni latenti. Modellando la distribuzione con GML, la rete riesce a catturare meglio le strutture complesse nei dati delle immagini portando ad una codifica entropica più accurata e riducendo la dimensione del file compresso.

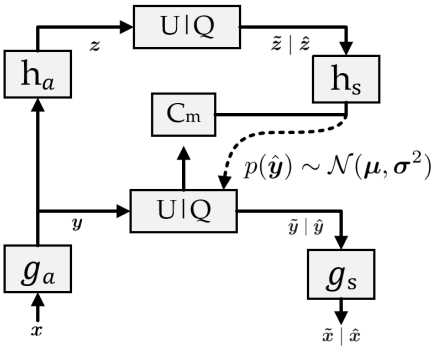
I ricercatori sono partiti considerando la ridondanza spaziale interna alle codifiche ottenute da altre tecniche recenti e ponendosi l'obiettivo di minimizzarla per poter codificare le immagini con meno bit e quindi ottenere tassi di compressione più elevati. Sebbene ci siano molti lavori, tra cui quelli presentati prima, che sperimentano con la compressione appresa, c'è ancora un divario tra le distribuzioni stimate e quelle reali per le rappresentazioni latenti.



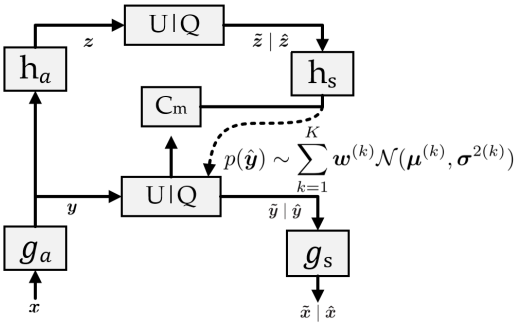
(a) Baseline



(b) Hyperprior model proposto da [36]



(c) Joint model proposto da [37]



(d) Modello proposto da [38]

Figura 5.1: Evoluzione dei modelli che hanno portato a [38]

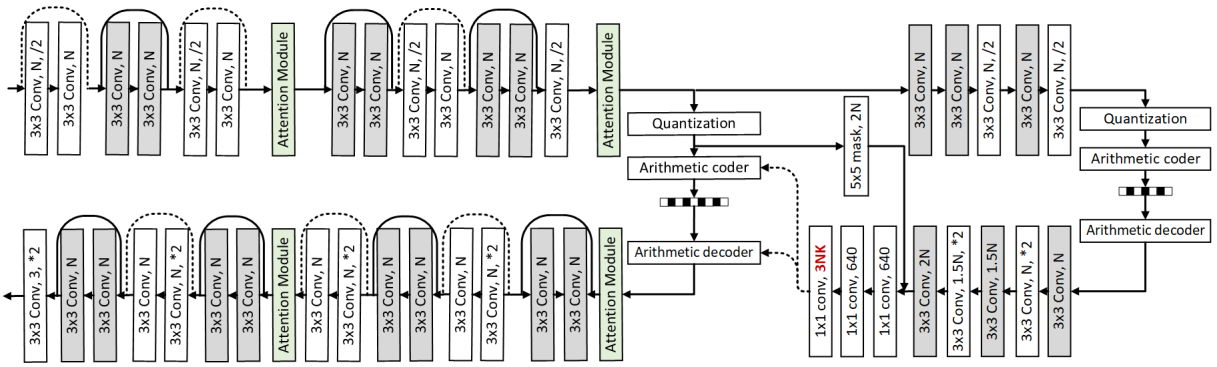


Figura 5.2: Architettura completa della rete

Il modello proposto è quello mostrato in figura 5.1d dove

x	è l'immagine originale
$y = g_a(x, \phi)$	è la rappresentazione latente della codifica prima della quantizzazione
$\hat{y} = Q(y)$	è la quantizzazione di y ovvero il codice compresso
$\hat{x} = g_s(\hat{y}, \theta)$	è l'immagine ricostruita
ϕ, θ	sono i parametri da ottimizzare per l'analisi e la sintesi dell'immagine
$z = h_a(y, \phi_h)$	è l'informazione aggiuntiva
$\hat{z} = Q(z)$	è la quantizzazione di z
$p_{\hat{y}\|\hat{z}}(\hat{y}\|\hat{z}) \leftarrow h_s(\hat{z}, \theta_h)$	è la stima della distribuzione condizionata su $\hat{z}$
ϕ_h, θ_h	sono i parametri ottimizzati per l'informazione aggiuntiva

Il modello a mescolanza Gaussiana nella sua formulazione continua è

$$p_{\hat{y}|\hat{z}}(\hat{y}|\hat{z}) \sim \sum_{k=1}^{K} w^{(k)} \mathcal{N}\left(\mu^{(k)}, \sigma^{2(k)}\right)$$

dove k è l'indice della componente Gaussiana, ma essendo $\hat{y}$ quantizzato e quindi discreto

$$p_{\hat{y}|\hat{z}}(\hat{y}|\hat{z}) = \prod_i p_{\hat{y}|\hat{z}}(\hat{y}_i|\hat{z})$$

$$p_{\hat{y}|\hat{z}}(\hat{y}_i|\hat{z}) = \left(\sum_{k=1}^{K} w_i^{(k)} \mathcal{N}\left(\mu_i^{(k)}, \sigma_i^{2(k)}\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)(\hat{y}_i)$$

dove i indica l'indice della posizione della mappa delle features. Sostanzialmente il modello riesce a raggiungere performance migliori in quanto seleziona i K valori più probabili. Gli esperimenti condotti fissano $K = 3$.

Il modello include anche dei moduli di attenzione (vedi figura 5.2) che permettono di individuare le zone dell'immagine più ricche di dettagli e che quindi richiedono un maggior numero di bit.

5.2 Training

Durante la fase di training è stato usato un segnale di rumore uniforme per simulare la quantizzazione. Per quanto riguarda il tasso invece si può usare l'entropia di y come stima del bitrate in quanto la codifica entropica aritmetica utilizzata è quasi ottima. Come loss function invece viene usata una metrica che tiene in considerazione sia il tasso complessivo dato da $\hat{y}$ e $\hat{z}$, che

la distorsione (quest'ultima attraverso il fattore lagrangiano λ che controlla il tradeoff) ovvero

$$\mathcal{L} = \mathcal{R}(\hat{y}) + \mathcal{R}(\hat{z}) + \lambda \mathcal{D}(x, \hat{x})$$

La fase di training è stata condotta su un subset di ImageNet ritagliato in campioni di 256×256 pixel e come metrica di distorsione MSE e MS-SSIM.

5.3 Risultati

I risultati riportati dai ricercatori per il modello allenato con MS-SSIM sono i migliori tra tutti gli altri modelli presentati se comparati tramite quella metrica e analogamente il modello allenato tramite MSE è tra i migliori per quanto riguarda il PSNR. In generale possiamo dire che l'allenamento con MS-SSIM ha dato risultati migliori, sia dal confronto con le altre tecniche che da un punto di vista percettivo rispetto a quello ottenuto con MSE.

I risultati da me ottenuti (vedi figura 5.3) differiscono parzialmente da quanto riportato nell'articolo. Ho testato MSE, PSNR e SSIM di JPEG, JPEG2000 e del modello proposto da Cheng allenato su MS-SSIM (attraverso le API disponibili presso [39]) in modo da confrontare l'architettura proposta con le più note tecniche di compressione tradizionale. I dataset analizzati sono Kodak [40] e Kitti [41]. Per ciascuna tecnica e ciascun dataset ho calcolato ciascuna metrica sia nello spazio di colori RGB che YCbCr, nel primo caso componendo i risultati dei tre canali con una semplice media aritmetica e nel secondo con una media ponderata 611.

L'esito delle misurazioni mostra chiaramente quanto la scelta di una metrica cambi il risultato percepito. A parità di immagine decodificata per ciascuna tecnica, le prestazioni misurate nello spazio YCbCr risultano sempre notevolmente migliori. Questo era ragionevolmente prevedibile per quanto riguarda le due codifiche tradizionali prese in esame secondo quanto detto al capitolo 3 mentre per quanto riguarda il modello di Cheng indica che probabilmente le metriche utilizzate per l'allenamento seguivano questa stessa convenzione sebbene non specificatamente indicato nell'articolo.

Un altro aspetto non chiaramente indicato nell'articolo è quello che riguarda i parametri per l'impostazione della codifica JPEG2000. I risultati da me riportati sono stati ottenuti utilizzando la libreria `openjpeg` in Python [42] impostando nella funzione `encode` i parametri

- `compression_ratios` pari ad un solo valore per ciascuna iterazione e variando il valore stesso da 10 a 200 con step di 10. In questo modo otteniamo 20 immagini con compression ratio da 10 a 200 e ciascuna con un solo quality layer

- `photometric_interpretation=1` per indicare che l'array ricavato dal file originario PNG è in formato sRGB e quindi per permettere una corretta interpretazione dei valori nel file JP2

Variando questi parametri, ad esempio impostando più di un quality layer per ciascuna immagine, si ottengono valori per le metriche molto diversi. Nel casi in questione addirittura per il PSNR risulta JPEG2000 migliore del modello di Cheng mentre per SSIM il vantaggio dell'architettura basata sulle reti neurali è più evidente.

Bisogna infine considerare che, sebbene il tempo di esecuzione sia fortemente dipendente dalla macchina su cui si eseguono le procedure di codifica e decodifica, il modello basato su reti neurali è sensibilmente più lento delle tecniche tradizionali, al punto da renderlo non utilizzabile per applicazioni pratiche.

In figura 5.4 e 5.5 si mostrano rispettivamente l'immagine originale tratta dal dataset Kodak e alcuni esempi di codifica con JPEG, JPEG2000 e il modello di Cheng a diversi livelli di qualità. In particolare si può notare come JPEG a bassi bitrate (figura 5.5a) generi degli artefatti molto visibili e fastidiosi: aloni, perdita di dettagli, blocking (cioè si riconoscono chiaramente i blocchi 8×8) e soprattutto la bassa profondità di colore che genera bande uniformi e separate al posto di transizioni fluide nelle campiture più grandi. Già raddoppiando il bitrate (figura 5.5b) tutti questi problemi sono sostanzialmente svaniti e la distorsione rispetto all'immagine originale è ridotta. A parità di bitrate, JPEG2000 porta ad una qualità percepita decisamente migliore (figura 5.5c). Si può notare una piccola perdita di dettagli e un leggero effetto blocking ma è assente l'effetto a bande di colore visto con JPEG. Anche a bitrate più alti (figura 5.5b) la qualità di JPEG2000 è migliore in quanto l'immagine risulta più morbida nelle campiture più grandi e più dettagliata nei soggetti in primo piano. L'effetto della codifica con Cheng2020 a bassissimo bitrate (figura 5.5e) è quello di rendere l'immagine molto morbida con i colori sfumati e dettagli mancanti ma allo stesso tempo con i bordi dei soggetti in primo piano ben delineati che la rendono nel complesso gradevole. Scegliendo invece la codifica ad alta qualità (figura 5.5f) si guadagna molto in termini di dettagli e nitidezza dell'immagine, che risulta indistinguibile dall'originale pur con un compression ratio considerevole.

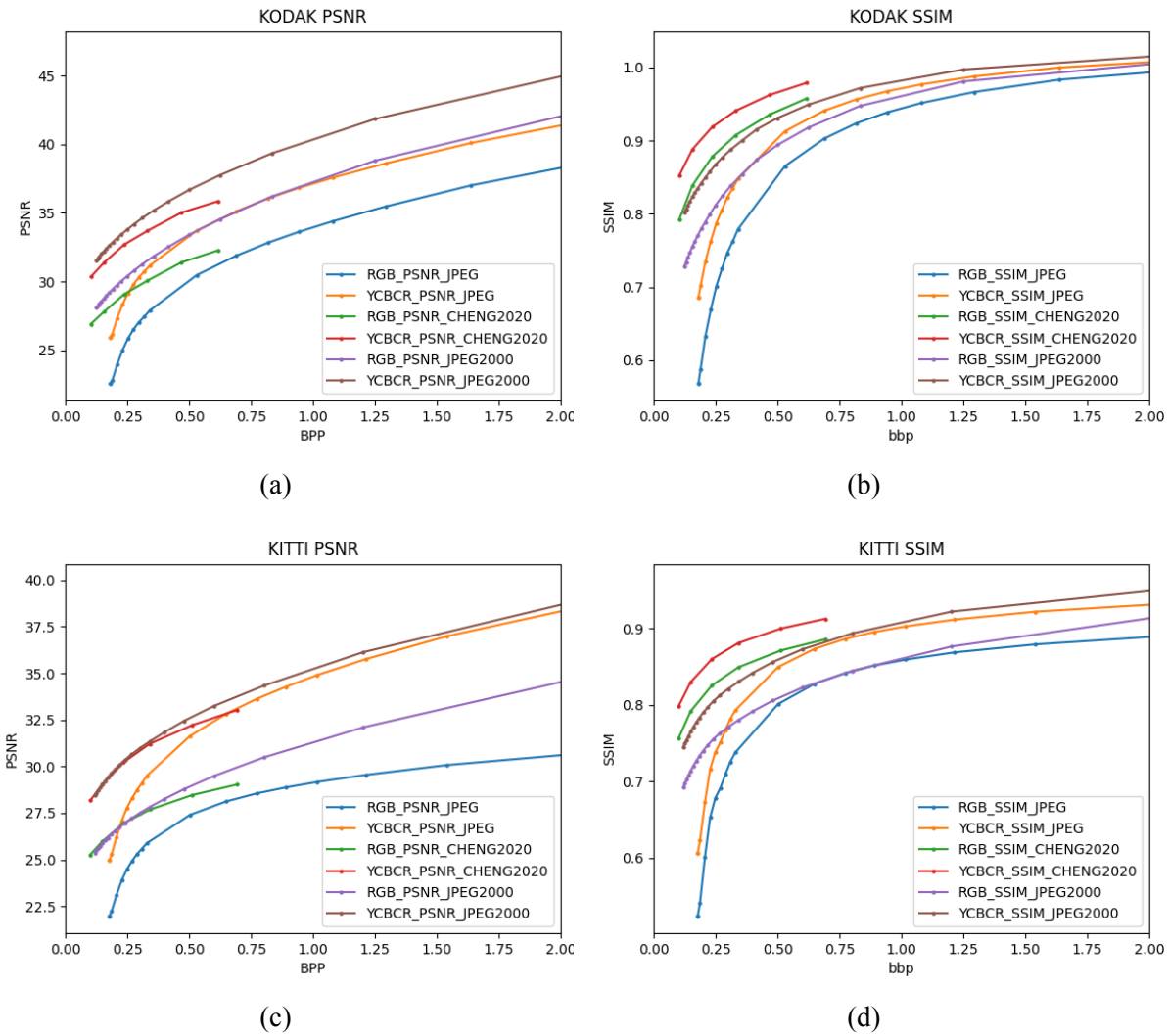


Figura 5.3: Risultati degli esperimenti da me condotti. In ascissa sempre il bitrate in bpp, in ordinata a sinistra il PSNR in dB e a destra il SSIM nell'intervallo $[0,1]$

Figura 5.4: Immagine originale codificata in PNG con 11.344 bpp

(a) Immagine codificata con JPEG (q=10 nel modulo Python `PIL`) con 0.237 bpp, CR=47.912, PSNR=32.449dB, SSIM=0.863

(b) Immagine codificata con JPEG (q=50 nel modulo Python `PIL`) con 0.565 bpp, CR=20.090, PSNR=38.418dB, SSIM=0.949

(c) Immagine codificata con JPEG2000 (q=100 nel modulo Python `openjpeg`) con 0.239 bpp, CR=47.278, PSNR=37.815dB, SSIM=0.933

(d) Immagine codificata con JPEG2000 (q=50 nel modulo Python `openjpeg`) con 0.479 bpp, CR=23.671, PSNR=41.233dB, SSIM=0.959

(e) Immagine codificata con Cheng2020 (q=1) con 0.067 bpp, CR=169.431, PSNR=32.021dB, SSIM=0.912

(f) Immagine codificata con Cheng2020 (q=6) con 0.354 bpp, CR=32.033, PSNR=38.221dB, SSIM=0.964

Figura 5.5: Esempio di immagini codificate con le varie tecniche e a vari tassi. Le metriche di PSNR e SSIM riportate sono calcolate nello spazio YCbCr con la media ponderata 611

Capitolo 6

Conclusioni

In questo elaborato sono state presentati i concetti principali relativi alle immagini digitali e alla loro compressione. In seguito sono state analizzate alcune delle più note metriche che permettono di valutare un sistema di compressione di immagini per poi passare alla trattazione di due tra i più noti sistemi di compressione tradizionali, JPEG e JPEG2000. Siamo passati poi allo studio delle reti neurali con alcune delle famiglie più importanti e relativi esempi di applicazione nell'ambito della compressione. Infine più nel dettaglio una tecnica recente e innovativa con risultati buoni, ma non rivoluzionari, secondo le metriche qualitative. Purtroppo la complessità della rete rende molto più lenta l'esecuzione rispetto alle tecniche tradizionali.

La parte sperimentale ha fatto emergere la difficoltà nell'instaurare dei confronti equi tra tecniche profondamente diverse e alcuni aspetti del modello proposto [38] che non erano stati sottolineati nell'articolo.

Negli ultimi anni è esplosa la ricerca nell'ambito dell'intelligenza artificiale e con essa anche i risultati rilevanti. La compressione di immagini con reti neurali vede molte architetture e tecniche proposte ma resta per ora un ambito di ricerca non destinato a rivoluzionare nel breve termine le tecnologie multimediali e di telecomunicazioni.

Bibliografia

[1] «Most valuable companies 2023,» Statista. (2023), indirizzo: https://www.statista.com/statistics/263264/top-companies-in-the-world-by-market-capitalization/ (visitato il 07/09/2024).

[2] «Companies ranked by market cap - CompaniesMarketCap.com.» (2024), indirizzo: https://companiesmarketcap.com/ (visitato il 07/09/2024).

[3] M. Roser, «The Internet's history has just begun,» *Our World in Data*, 2018. indirizzo: https://ourworldindata.org/internet-history-just-begun.

[4] Cisco Systems, «Cisco annual internet report (2018–2023) white paper,» Cisco Systems, rapp. tecn., 2020. indirizzo: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf (visitato il 11/08/2024).

[5] S. Sherrington, «Scaling to 800g in operator metro core, backbone and DCI networks,» Analysys Mason, rapp. tecn., 2023. indirizzo: https://www.analysysmason.com/contentassets/9c29db814d7743e6a5953d55dff65885/analysys_mason_800g_operator_networks_feb2023_rma22.pdf (visitato il 07/09/2024).

[6] C. E. Shannon, «A mathematical theory of communication,» *The Bell System Technical Journal*, vol. 27, n. 3, pp. 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[7] W. Commons. «Pixel geometry 01 Pengo.» (2007), indirizzo: https://commons.wikimedia.org/wiki/File:Pixel_geometry_01_Pengo.jpg (visitato il 13/09/2024).

[8] Wikimedia Commons. «Common Chroma Subsampling Ratios (YCbCr).» (2024), indirizzo: https://commons.m.wikimedia.org/wiki/File:Common_chroma_subsampling_ratios_YCbCr_CORRECTED.svg (visitato il 14/09/2024).

[9] L.-H. Chen, C. G. Bampis, Z. Li, J. Sole e A. C. Bovik, «Perceptual Video Quality Prediction Emphasizing Chroma Distortions,» *IEEE Transactions on Image Processing*, vol. 30, pp. 1408–1422, 2021. doi: 10.1109/TIP.2020.3043127.

[10] Z. Wang, A. Bovik, H. Sheikh e E. Simoncelli, «Image quality assessment: from error visibility to structural similarity,» *IEEE Transactions on Image Processing*, vol. 13, n. 4, pp. 600–612, 2004. doi: 10.1109/TIP.2003.819861.

[11] Z. Wang, E. Simoncelli e A. Bovik, «Multiscale structural similarity for image quality assessment,» in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, 2003, 1398–1402 Vol.2. doi: 10.1109/ACSSC.2003.1292216.

[12] JPEG. «JPEG 2000 logo.» ver. 1.0. (2023), indirizzo: https://jpeg.org/images/jpeg-logo.svg (visitato il 18/09/2024).

[13] G. Wallace, «The JPEG still picture compression standard,» *IEEE Transactions on Consumer Electronics*, vol. 38, n. 1, pp. xviii–xxxiv, 1992. doi: 10.1109/30.125072.

[14] N. Ahmed, T. Natarajan e K. Rao, «Discrete Cosine Transform,» *IEEE Transactions on Computers*, vol. C-23, n. 1, pp. 90–93, 1974. doi: 10.1109/T-C.1974.223784.

[15] User: Alex Khristov. «ZigZag Scan.» (2007), indirizzo: https://it.m.wikipedia.org/wiki/File:JPEG_ZigZag.svg.

[16] User: Lulu of the Lotus-Eaters. «File:Sego lily cm-150.png.» (2006), indirizzo: https://commons.wikimedia.org/wiki/File:Sego_lily_cm-150.png.

[17] User: Lulu of the Lotus-Eaters. «File:Sego lily cm-150.jpg.» (2006), indirizzo: https://commons.wikimedia.org/wiki/File:Sego_lily_cm-150.jpg.

[18] JPEG. «JPEG 2000 logo.» ver. 1.0. (2023), indirizzo: https://jpeg.org/images/jpeg2000-logo.svg (visitato il 18/09/2024).

[19] D. Taubman e M. Marcellin, «JPEG2000: standard for interactive imaging,» *Proceedings of the IEEE*, vol. 90, n. 8, pp. 1336–1357, 2002. doi: 10.1109/JPROC.2002.800725.

[20] JPEG.org. «JPEG 2000 Applications.» (2024), indirizzo: https://jpeg.org/jpeg2000/applications.html (visitato il 18/09/2024).

[21] W. S. McCulloch e W. Pitts, «A logical calculus of the ideas immanent in nervous activity,» *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[22] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*, https://bpb-us-e2.wpmucdn.com/websites.umass.edu/dist/a/27637/files/2016/03/rosenblatt-1957.pdf, 1957.

[23] User: Omegatron. «Dirac distribution CDF.» (2006), indirizzo: https://commons.wikimedia.org/wiki/File:Dirac_distribution_CDF.svg.

[24] User: Qef. «Logistic-curve.» (2008), indirizzo: https://commons.wikimedia.org/wiki/File:Logistic-curve.svg.

[25] F. Rosenblatt, *Perceptions and the theory of brain mechanisms*. Spartan books, 1962.

[26] User: Wiso. «Neural network example.» (2008), indirizzo: https://commons.wikimedia.org/wiki/File:Neural_network_example.svg.

[27] J. Johnson. «What's a Deep Neural Network? Deep Nets Explained.» (2020), indirizzo: https://www.bmc.com/blogs/deep-neural-network/ (visitato il 19/09/2024).

[28] Sonehara, Kawato, Miyake e Nakane, «Image data compression using a neural network model,» in *International 1989 Joint Conference on Neural Networks*, 1989, 35–41 vol.2. doi: 10.1109/IJCNN.1989.118675.

[29] I. S. Mohamed, «Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques,» tesi di dott., set. 2017. doi: 10.13140/RG.2.2.30795.69926.

[30] J. Ballé, V. Laparra e E. Simoncelli, «End-to-end Optimized Image Compression,» nov. 2016. doi: 10.48550/arXiv.1611.01704.

[31] User: Ixnay. «Recurrent neural network unfold.» (2017), indirizzo: https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg.

[32] S. Hochreiter e J. Schmidhuber, «Long Short-Term Memory,» *Neural Computation*, vol. 9, n. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[33] A. Zhang, Z. C. Lipton, M. Li e A. J. Smola. «Computing the hidden state in an LSTM model.» (2021), indirizzo: https://d2l.ai/chapter_recurrent-modern/lstm.html (visitato il 21/09/2024).

[34] G. Toderici, D. Vincent, N. Johnston et al., «Full Resolution Image Compression with Recurrent Neural Networks,» in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5435–5443. doi: 10.1109/CVPR.2017.577.

[35] O. Rippel e L. Bourdev, *Real-Time Adaptive Image Compression*, 2017. arXiv: 1705.05823 [stat.ML]. indirizzo: https://arxiv.org/abs/1705.05823.

[36] J. Ballé, D. Minnen, S. Singh, S. J. Hwang e N. Johnston, *Variational image compression with a scale hyperprior*, 2018. arXiv: 1802.01436 [eess.IV]. indirizzo: https://arxiv.org/abs/1802.01436.

[37] D. Minnen, J. Ballé e G. Toderici, *Joint Autoregressive and Hierarchical Priors for Learned Image Compression*, 2018. arXiv: 1809.02736 [cs.CV]. indirizzo: https://arxiv.org/abs/1809.02736.

[38] Z. Cheng, H. Sun, M. Takeuchi e J. Katto, *Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules*, 2020. arXiv: 2001.01568 [eess.IV]. indirizzo: https://arxiv.org/abs/2001.01568.

[39] InterDigitalInc. «CompressAI.» Accessed: 2024-09-21. (2024), indirizzo: https://github.com/InterDigitalInc/CompressAI.

[40] E. K. Company, *Kodak Lossless True Color Image Suite*, Accessed: 2024-09-21, 1993. indirizzo: http://r0k.us/graphics/kodak/.

[41] A. Geiger, P. Lenz, C. Stiller e R. Urtasun, *The KITTI Vision Benchmark Suite*, Accessed: 2024-09-21, 2012. indirizzo: http://www.cvlibs.net/datasets/kitti/.

[42] pydicom. «pylibjpeg-openjpeg.» Accessed: 2024-09-21. (2024), indirizzo: https://github.com/pydicom/pylibjpeg-openjpeg/.