Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Rimozione di artefatti da movimento da immagini

Valutazione di immagini tomografiche con reti neurali

Relatore:
Chiar.ma Prof.ssa
Elena Loli Piccolomini

Presentata da:
Andrea Valente

II sessione
Anno Accademico: 2020/2021

Introduzione

La sfocatura da movimento è un problema diffuso in diversi tipi di immagini. Per fare un esempio, si può riscontrare nella fotografia, quando le persone o gli oggetti che si vogliono catturare si muovono, oppure in altre tipologie di campi. In questo lavoro, verranno trattati i risultati che si hanno quando la sfocatura avviene nelle immagini mediche ottenute nella tomografia computerizzata.

Queste sono ricostruite attraverso una retro-proiezione dei *raggi X*, che sono disposti a diversi angoli attorno l'oggetto che si vuole esaminare. Inoltre, i raggi che sono emessi non hanno un percorso netto dal proiettore al ricevitore, ma sono sempre influenzati da rifrazione e diffrazione, che comportano del rumore nella tomografia che si vuole ottenere.

A queste immagini, inoltre, si aggiunge il problema della sfocatura causata, per esempio, dal movimento del paziente durante la scansione, per la quale sono necessari anche diversi minuti per potersi completare.

Oltre a questo, con la diffusione dell'apprendimento profondo — *Deep Learning* —, negli ultimi anni la ricostruzione delle immagini ha avuto progressi importanti. Per questo motivo è stata utilizzata la rete neurale *DeblurGAN-v2*, considerata lo stato dell'arte nella rimozione della sfocatura da immagini.

Per una spiegazione dettagliata del caso di studio, il problema verrà illustrato dividendolo in tre macro-aree:

1. La convoluzione, il suo problema inverso, ovvero la deconvoluzione, le sue applicazioni e, infine, un approfondimento sulla sfocatura e le sue tipologie.

2. La rete DeblurGAN-v2, dando importanza alla sua architettura e ai dataset utilizzati durante l'apprendimento, per sottolineare come le immagini mediche usate hanno caratteristiche differenti rispetto a quelle impiegate dagli autori.

3. Infine, i risultati numerici in base a diversi parametri, per esempio, aggiungendo del rumore gaussiano per verificare come la rete si comporta in questo caso. Saranno, inoltre, valutate le immagini anche in base agli artefatti eliminati durante il processo di ricostruzione e verrà fatto un confronto con un altra rete, *SelfDeblur*.

Indice

Capitolo 1

Problema di convoluzione e artefatti da movimento

Nel campo dell'elaborazione delle immagini digitali, l'operazione di *convoluzione* è molto diffusa, infatti, trova risonanza in numerosi campi, soprattutto nella fotografia.

Più interessante come fenomeno di studio è la sua operazione inversa, ossia la *deconvoluzione*, che sarà l'argomento principale di questo lavoro e di cui si cercheranno diverse metodologie per affrontarla nel contesto dell'eliminazione della sfocatura da movimento nelle immagini.

1.1 Convoluzione lineare

La *convoluzione lineare* riguarda l'applicazione, ad un'immagine, di un filtro detto *kernel* di convoluzione: in questo modo si otterrà una nuova immagine. Quest'ultima differirà dall'originale secondo alcune caratteristiche che verranno esaltate dal kernel preso in considerazione.

Possiamo descrivere il problema con la seguente equazione:

$$y = x \otimes k \tag{1.1}$$

dove y è l'immagine ottenuta in seguito alla convoluzione, x è l'immagine di partenza e, infine, k è il kernel applicato all'immagine.

Nel caso reale, oltre al kernel, è sempre presente un rumore additivo η, che può provenire da varie fonti di tipo ambientale durante l'acquisizione dell'immagine come, per esempio, dal livello di luce.

Riassumendo, il problema generale è posto come segue:

$$y = x \otimes k + \eta \tag{1.2}$$

Scendendo nei dettagli, il motivo della convoluzione è quello di ottenere un'immagine in cui ogni pixel viene modificato tenendo in considerazione anche quelli vicini.

In particolare, il filtro che verrà usato, di solito di piccola dimensione, viene centrato su ogni pixel dell'immagine e, a questo punto, avviene l'operazione di convoluzione in cui si moltiplica ogni pixel del kernel con il rispettivo pixel su cui è posizionato e i valori ottenuti vengono sommati tra loro e normalizzati. In questo modo il valore della luminosità nell'immagine ottenuta rimane lo stesso rispetto all'immagine originale.

A questo punto si può notare un comportamento particolare lungo i bordi dell'immagine, dato che il kernel avrebbe alcuni valori fuori da essa. In generale, si può procedere in varie direzioni:

- Non considerare i pixel che si trovano fuori dall'immagine.
- Aggiungere dei pixel lungo il bordo dell'immagine attraverso la tecnica *padding*. I pixel aggiunti potrebbero replicare quelli più vicini oppure avere un valore costante, come l'essere neri.
- Escludere il kernel qualora finisse fuori dall'immagine.

Solo nell'ultimo caso avremmo un'immagine y di dimensione ridotta rispetto a quella di partenza. Infatti, fissando l'immagine originale x di dimensione $m \times n$ e il kernel k di dimensione $p \times q$ (con p e q dispari), viene generata un'immagine y di dimensione $(m - p + 1) \times (n - q + 1)$.

In generale, l'operazione di convoluzione consiste nel calcolo di in ogni pixel dell'immagine y come segue:

$$y(i, j) = \sum_{u=1}^{p} \sum_{v=1}^{q} x(i + u - 1, j + v - 1) * k(u, v) \tag{1.3}$$

1.2 Deconvoluzione

Molto più interessante ai fini del lavoro è la deconvoluzione — *deconvolution*. È l'operazione inversa a quella di convoluzione: a partire da un'immagine convoluzionata, l'intento è quello di ottenere quella originale.

Possiamo distinguere due metodologie per questa operazione: la *non-blind deconvolution* oppure la *blind deconvolution*. Esse differiscono dalle informazioni che si hanno a disposizione per ricostruire l'immagine di partenza, infatti, nella prima si conosce sia l'immagine filtrata y che il kernel k, mentre nella seconda l'unico termine conosciuto è y.

Per quanto la prima tecnica permetta di ricavare correttamente l'immagine di partenza, avendo solo un'incognita e due termini noti, non trova impiego nella maggior parte dei campi dato che il kernel non è quasi mai conosciuto. In conclusione, il problema di *blind deconvolution* è un problema mal posto, in cui conoscendo y dobbiamo ricavare sia k che x: questo significa che esistono infinite coppie, formate da un'immagine e un filtro, che calcolano la stessa immagine. Oltre a ciò, l'immagine filtrata è compromessa da rumore additivo, complicando ulteriormente il problema di poter ricavare l'originale senza artefatti[1].

1.3 Utilizzi della convoluzione

La convoluzione, come discusso in § 1.1, trova risalto in un gran numero di campi. Considerando che il lavoro di questa tesi si concentrerà sull'elaborazione di immagini, verranno passate in rassegna alcune applicazioni che la riguardano più da vicino. Tra cui:

- *Smoothing*: detto anche *lisciamento*, consiste nel calcolo di ogni pixel della nuova immagine come una media tra le intensità degli altri pixel nelle vicinanze, incluso se stesso. Viene utilizzato spesso per ridurre il rumore nelle immagini.
- *Edge detection*: è un filtro che va a rilevare i bordi grazie ai cambi di intensità che avvengono nell'immagine.

- *Sharpening*: permette di aumentare i dettagli di un'immagine mettendo più in risalto il pixel centrale aumentandone l'intensità.

Di seguito verrà introdotto un particolare filtro di *smoothing*, ossia la sfocatura. In questo caso, oltre alle principali caratteristiche del lisciamento, avviene anche una perdita di dettagli nell'immagine ottenuta.

1.3.1 Fitro di sfocatura

La sfocatura, detta anche *blur*, è una particolare operazione di convoluzione in cui viene applicato un kernel di sfocatura all'immagine, ottenendone, quindi, una sfocata come risultato (in Figura 1.1 è possibile vedere un esempio).

Solitamente la causa della sfocatura nelle immagini è data da soggetti in movimento che vengono catturati da video o immagini scattate con un alto tempo di esposizione, oppure dal movimento della mano che tiene la camera nel momento dello scatto.

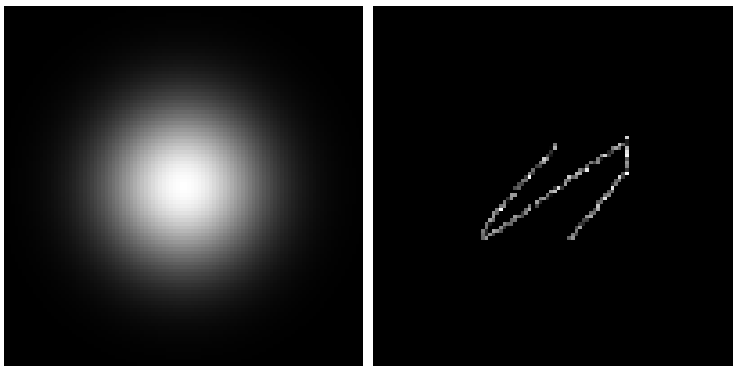
(a) Immagine originale (b) Filtro (c) Immagine sfocata

Figura 1.1: Esempio di convoluzione con filtro di sfocatura[2]

Questo kernel di blur può essere di due tipi:

- *Sfocatura Gaussiana*: è il filtro più utilizzato per ottenere un effetto di *blur* sull'immagine originale. I valori del kernel sono calcolati secondo la distribuzione normale, di conseguenza, il pixel calcolato conterrà sempre meno informazioni allontanandosi da quello centrale (Figura 1.2a), come una funzione Gaussiana.

- *Sfocatura da movimento*: il *motion blur*, a differenza del filtro precedente, si trova in situazioni reali, come nel caso di oggetti in movimento. Per questo la sfocatura ha caratteristiche differenti da immagine a immagine che difficilmente sono catturabili. In questo caso, il filtro generato contiene dei valori lungo una particolare direzione (Figura 1.2b), quella del movimento e non di un cerchio, come nella sfocatura Gaussiana.



(a) Blur gaussiano (b) Blur di movimento

Figura 1.2: Tipologie di sfocatura

Oltre a questa differenza tra le due tipologie di sfocatura, il filtro può essere applicato in due modi differenti all'immagine, producendo risultati diversi. Il primo caso è quello della sfocatura *uniforme*, quella in cui lo stesso kernel viene applicato uniformemente sull'immagine. Il caso contrario, di sfocatura *non uniforme*, è quello più comune nelle immagini reali degli oggetti in movimento, in quanto ognuno ha una propria direzione.

Quest'ultimo caso è molto più complesso e deve essere tenuto in considerazione durante la risoluzione del problema, diversamente non sarebbero raggiunti i risultati sperati.

1.4 Blind deconvolution con Deep Learning

La risoluzione del problema di deconvoluzione *blind* che prevede filtri di sfocatura rimane difficile da risolvere, specialmente per le immagini reali che hanno kernel difficili da stimare. Resta comunque il fatto che, negli anni, sono state sviluppate diverse tecniche

per avvicinarsi a un'immagine $\hat{x}$ che sia riconducibile all'immagine originale x, con un minor numero di artefatti possibili in cui si possono distinguere alcuni dettagli salienti che altrimenti sarebbero persi nell'immagine affetta da sfocatura.

Fino a qualche anno fa, le principali proposte erano i metodi di ottimizzazione, specialmente i metodi di regolarizzazione come, per esempio, la stima del massimo a posteriori — *maximum a posteriori estimation*. Questi, però, non riuscivano a raggiungere una vera e propria ricostruzione che fosse quanto meno fedele all'immagine reale, soprattutto nei casi di sfocatura non uniforme.[3] [4]

Negli ultimi anni, quindi, sono state sperimentate nuove tecniche nella frontiera dell'elaborazione di immagini, avvicinandosi alle tecniche basate su *Deep Learning*, raggiungendo risultati superiori rispetto a prima, specialmente per quello che riguarda una maggiore qualità percettiva.[5]

Uno dei problemi con le reti neurali addrestate è riconducibile alla fase di apprendimento. Se viene utilizzato un dataset composto da immagini sfocate fittizie, generate applicando all'immagine nitida un filtro di *blur*, la stima del kernel sarà molte volte sbagliata, di conseguenza, la rete non riuscirà a generalizzare la rimozione di artefatti ad altre tipologie di immagini che soffrono di una sfocatura più reale.[6]

Di seguito, verranno discusse le due principali tecniche che vengono adottate nello sviluppo di reti neurali per il problema dell'eliminazione del *motion blur* da un'immagine.

1.4.1 Kernel estimation

Questa tecnica è stata una delle prime sperimentate e una delle più efficaci per molti anni. Consiste nel risolvere il problema prima ricavando il kernel e poi, conoscendo sia y che k, applicando l'operazione di *non-blind deconvolution* ottenendo, così, l'immagine nitida. A eccezione della presenza di rumore nell'immagine con gli artefatti di movimento, i risultati potrebbero essere anche accurati rispetto all'immagine di *ground truth*.

La parte centrale di questa tecnica risiede nella stima del kernel e questa potrebbe complicarsi se si tratta di sfocatura non uniforme su tutta l'immagine.

Per questo, solitamente, si fa riferimento a delle tecniche di analisi dette *patch-wise*, in cui si stima un kernel differente per quelli che potrebbero essere i diversi oggetti in movimento. Questo permette di calcolare delle immagini più fedeli possibili a quelle

ricercate[5]. Il lato negativo di questa soluzione è l'alta complessità computazionale nell'individuare gli oggetti dell'immagine.

1.4.2 Image-to-image regression

Questa tecnica differisce dalla precedente perché non viene calcolato il kernel di convoluzione, ma viene stimata direttamente l'immagine senza sfocatura a partire da quella artefatta.

Le reti che sfruttano questa metodologia sono le ultime sviluppate e, nonostante richiedano uno sviluppo più complesso, permettono di ottenere tempi di computazione nettamente inferiori rispetto alle tecniche di *kernel estimation*, con risultati migliori.[6]

Capitolo 2

GAN e DeblurGANv2

Nel campo dell'elaborazione di immagini, le reti neurali hanno avuto maggiore visibilità grazie alla flessibilità con cui sono realizzate. Una particolare architettura di rete ha visto il suo utilizzo sempre di più aumentato, in quanto riesce meglio di altre nella generazioni di immagini: le reti generative avversarie, o *GAN*[7].

2.1 Generative Adversarial Network

Le reti generative avversarie sono contraddistinte da due particolari elementi: un generatore e un discriminatore. Entrambi sono associati a una funzione obiettivo che ha lo scopo di farli competere tra loro, questo viene menzionato come *min-max game*.

Scendendo nei dettagli, il discriminatore ha il compito di valutare se il dato in input, nel nostro caso un'immagine, sia reale oppure no. Dall'altra parte, il generatore genera delle immagini fittizie nella stessa distribuzione di quelle reali del *dataset*, cercando quindi di ingannare il discriminatore. La convergenza della rete è dettata dal raggiungimento di un equilibrio tra la funzione obiettivo del generatore e quella del discriminatore.

La funzione appena citata spesso è nella forma come segue:

$$\begin{aligned} \min_G \max_D V(D,G) =& \mathbb{E}_{x\sim pdata(x)}[logD(x)] \\ &+ \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))] \end{aligned} \tag{2.1}$$

2.2 DeblurGAN-v2

Il progetto di DeblurGAN-v2[8] è la seconda versione della rete neurale sviluppata sempre dagli stessi autori dal nome DeblurGAN[9]. Sono state introdotte interessanti novità sia rispetto al progetto precedente che per quanto riguarda l'attuale stato dell'arte della rimozione del *motion blur*.

2.2.1 Architettura

Il generatore utilizza la *Feature Pyramid Network*, una rete sviluppata nel campo del rilevamento di oggetti, per la prima volta nella ricostruzione delle immagini.

La pipeline (Figura 2.1) della rete è composta da diversi livelli che hanno lo scopo di estrarre le *feature* — informazioni che identificano dei pattern nell'immagine —, aumentare la risoluzione e rilevare gli oggetti. In questa maniera si ottengono cinque *feature map*, che vengono aumentate alla stessa di dimensione di $\frac{1}{4}$ di quella dell'input e sono concatenate in un tensore.

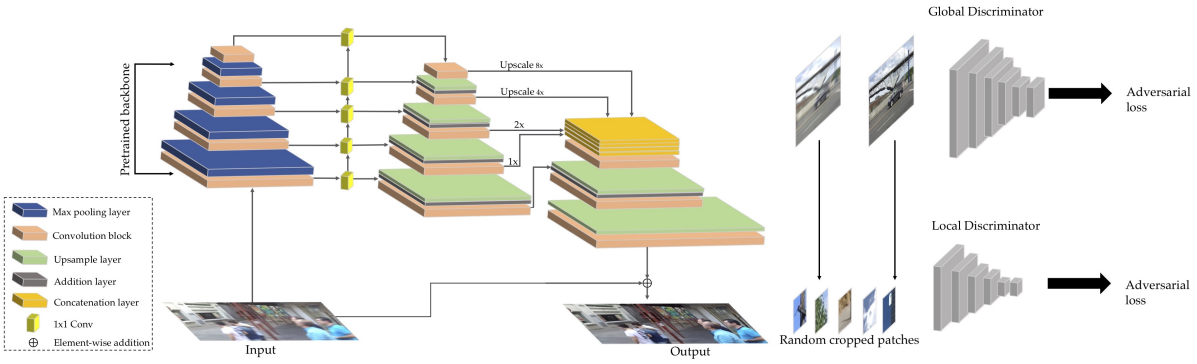


Figura 2.1: Pipeline

La flessibilità di questa rete sta nella scelta del modello di rete pre-allenato da utilizzare. In particolare, gli autori della rete hanno individuato due modelli che permettono di ottenere un *tradeoff* tra prestazioni e qualità della ricostruzione:

- *Inception-ResNet-v2*: raggiunge una maggiore accuratezza nell'immagine generata dalla rete.

- *MobileNet*: ha una costo computazionale inferiore ma con una ricostruzione dell'immagine con risultati inferiori rispetto al modello precedente.

La funzione obiettivo implementata differisce da quella dell'Equazione 2.1, per via del problema della *scomparsa del gradiente* (*vanishing gradient*) durante l'allenamento della rete profonda[10], per questo è stata adottata la funzione *Least Squares GAN*, definita come segue:

$$\begin{aligned} \min_D V(D) =& \frac{1}{2}\mathbb{E}_{x\sim pdata(x)}[(D(x)-1)^2] \\ &+ \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[D(G(z))^2] \end{aligned} \tag{2.2}$$

$$\min_D V(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-1)^2]$$

2.2.2 Training dataset

Per l'apprendimento della rete sono state considerate circa 10.000 immagini, scelte dai tre dataset più utilizzati nelle altre reti neurali e, quindi, più fedeli a casi di sfocatura reali.

Per tutti i casi, le immagini affette da artefatti da movimento sono ottenute tramite la tecnica di *averaging*, in cui ogni immagine del dataset è ottenuta come la media tra una serie di frame consecutivi di un video. Di seguito, una breve descrizione di ogni dataset:

- *GoPro*[6]: è composto da 3214 coppie di immagini nitide e mosse ottenute da sequenze di video di 240 frame al secondo di una GoPro Hero 4.

- *DVD*[11]: in questo caso le immagini sono ottenute da video di diversi dispositivi. In totale vengono generate 6708 coppie di immagini nitide e mosse.

- *NFS*[12]: contiene delle immagini eterogenee tra loro, provenienti da diversi ambienti reali, specialmente di carattere sportivo.

2.2.3 Valutazione delle prestazioni

Come si può vedere in Figura 2.2, i risultati ottenuti utilizzando i dataset di GoPro, Kohler e DVD dimostrano che la rete ottiene immagini con minori artefatti da movimento con miglioramenti pari o superiori a modelli utilizzati attualmente, come la rete SRN[13], ma in un tempo di computazione nettamente inferiore[8]. Questo dimostra come la rete possa permettere di essere usata anche in campi più avanzati come la rimozione della sfocatura da video in tempo reale.

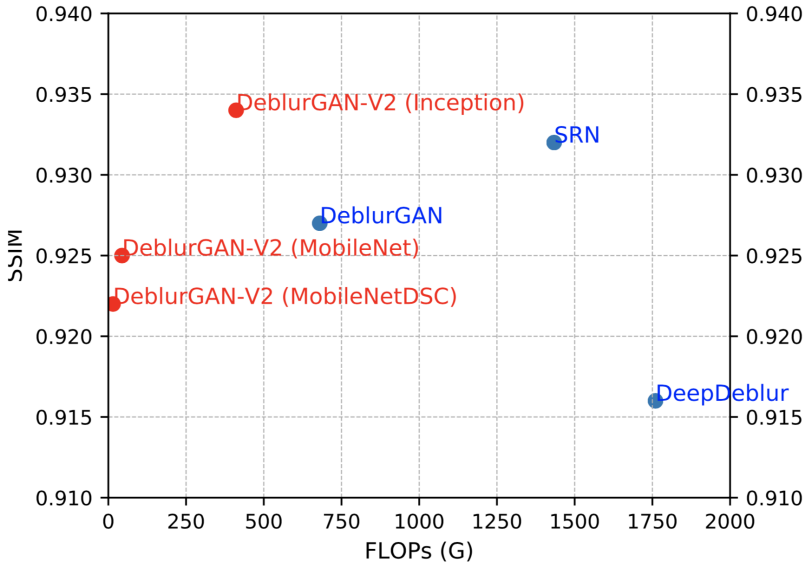


Figura 2.2: Confronto SSIM-FLOP tra metodi dello stato dell'arte

Il problema generale di queste reti è che spesso vengono considerati dataset poco eterogenei e, quindi, la capacità di generalizzare le immagini di varia natura spesso porta a risultati indesiderati. Inoltre, le metriche utilizzate per valutare le reti (*PSNR* e *SSIM*) non tengono conto della fedeltà percettiva, ancora molto difficile da misurare, per quanto alcune cerchino di avvicinarvisi.

Capitolo 3

Risultati numerici

In questo capitolo verranno discussi i risultati ottenuti dalla rete sul dataset preso in considerazione. Inoltre, verranno confrontati i dati con la rete SelfDeblur[14] che rappresenta l'attuale stato dell'arte per quanto riguarda la *blind deconvolution* di immagini affette da sfocatura da movimento.

Come anticipato, il dataset che sarà oggetto di test della rete contiene immagini mediche, in particolare, quelle ottenute con la tomografia computerizzata — *Computed Tomography*[15][16].

È possibile fare una valutazione quantitativa per i risultati che vengono ottenuti dalla rete neurale *DeblurGAN-v2*, grazie alle immagini di *ground truth*, dato che la sfocatura viene aggiunta manualmente costruendo un kernel di convoluzione.

Per questo motivo vengono utilizzate due metriche molto diffuse per confrontare le immagini:

- PSNR: *Peak signal-to-noise ratio.* È il rapporto tra l'intensità dell'immagine nitida e il *mean squared error* tra l'immagine con artefatti e l'immagine originale. In altre parole, calcola l'errore nel segnale affetto da rumore (Equazione 3.1).

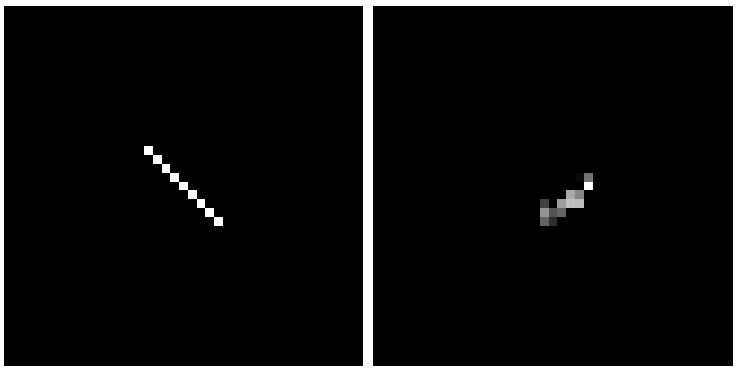
- SSIM[17]: *Structural Similarity.* A differenza della metrica precedente, che viene solitamente utilizzata per la sua facilità nel calcolo, questa tiene in considerazione anche la qualità percettiva delle immagini che vengono valutate garantendo un'affidabilità migliore nella valutazione.

$$
\begin{aligned}
MSE(GT, P) &= \frac{(GT - P)^2}{m * n} \\
PSNR(GT, P) &= 10 * log_{10}(\frac{MAX(GT)^2}{MSE(GT, P)})
\end{aligned}
\tag{3.1}
$$

Oltre a queste due metriche, ho fatto una valutazione soggettiva per individuare meglio quali sono le aree in cui l'eliminazione di artefatti è stata più accurata.

I confronti verranno fatti su due tipologie differenti di kernel di convoluzione:

- Kernel *lineare*: viene utilizzato un filtro di dimensione $9x9$ con valori non nulli lungo l'intera diagonale, come si può vedere in Figura 3.1a.
- Kernel di *spline*[18]: rispetto al filtro precedente, quest'altro può essere considerato un kernel più reale, dato che riproduce movimenti casuali. Per riuscirci, è stato implementato con una funzione di *spline* con quattro punti di controllo in una griglia di dimensione $8x8$, come in Figura 3.1b. È stato preso in considerazione un kernel con un movimento molto simile a quello lineare.



(a) Kernel lineare (b) Kernel spline

Figura 3.1: Kernel utilizzati

Inoltre, verranno valutate le immagini sia senza rumore che quelle con un rumore additivo guassiano con una deviazione standard pari al 5% rispetto a quella dell'immagine.

3.1 Valutazioni del dataset

Partendo dal filtro lineare, si può notare come i valori di PSNR e di SSIM aumentino dall'immagine con blur a quella ottenuta dalla rete, sia nei casi senza che con rumore additivo. Questo ci permette di valutare in maniera positiva il lavoro della rete nella ricostruzione delle immagini contenute nel dataset ottenuto con il kernel.

Considerando la metrica di SSIM (Figura 3.2), solitamente la più indicata per una miglior valutazione, si può notare come il valore mediano e, in generale la distribuzione dei valori, aumenti posizionandosi sopra il valore di 0,9, mentre in precedenza, nell'immagine con blur, era di circa di 0,87.

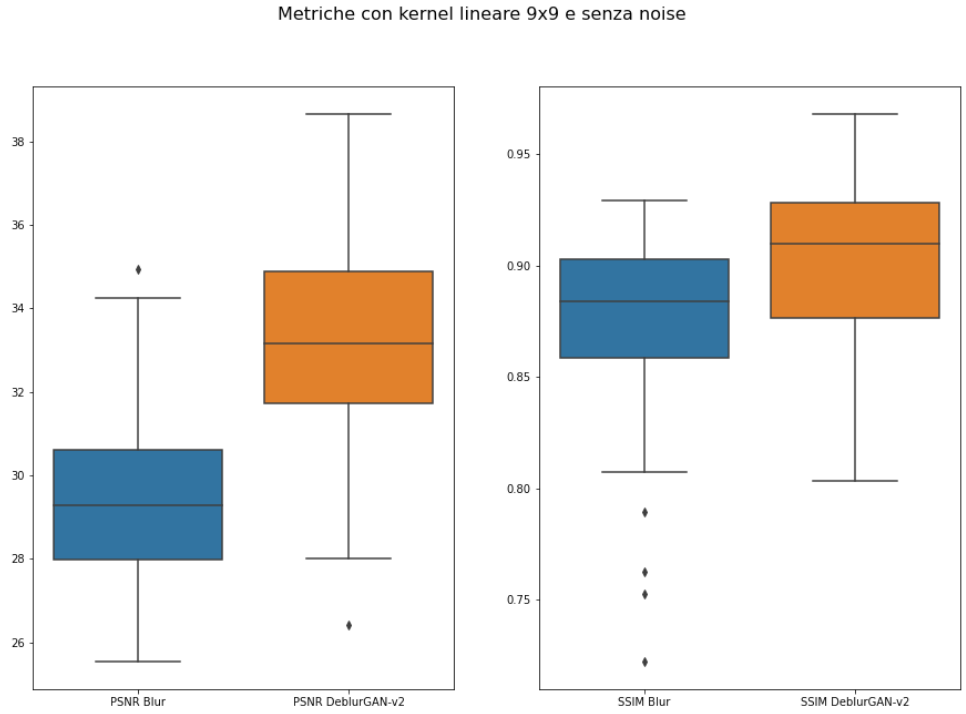
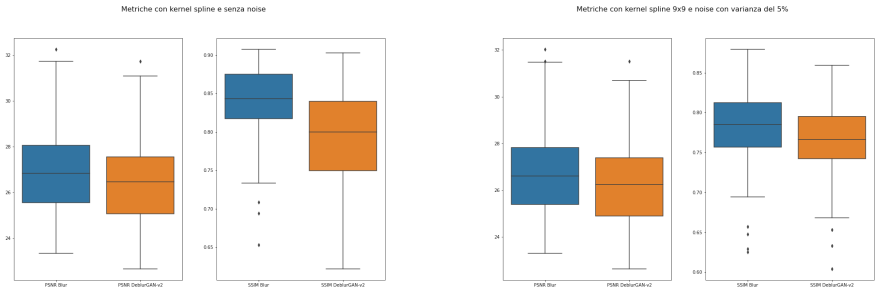


Figura 3.2: Box-plot kernel lineare senza rumore

Osservando invece i dati ottenuti con il kernel di spline si possono notare risultati meno scontati rispetto al kernel lineare. Infatti, per entrambe le metriche, i valori diminuiscono dalle immagini con blur a quelle ottenute dalle rete: si potrebbe intuire un

degrado nell'immagine ricostruita, ma ciò, invece, non accade e sarà approfondito meglio in § 3.2.

Una motivazione è data dalla particolarità del dataset preso in considerazione. Trattandosi di immagini mediche, sono i particolari quelli che contano di più, per questo la valutazione con queste metriche potrebbe trarre in inganno, dato che non tengono in considerazione dettagli come particolari di un osso o di un nervo.



(a) Senza rumore (b) Con rumore gaussiano del 5%

Figura 3.3: Box-plot kernel spline

3.2 Valutazione dell'eliminazione di artefatti

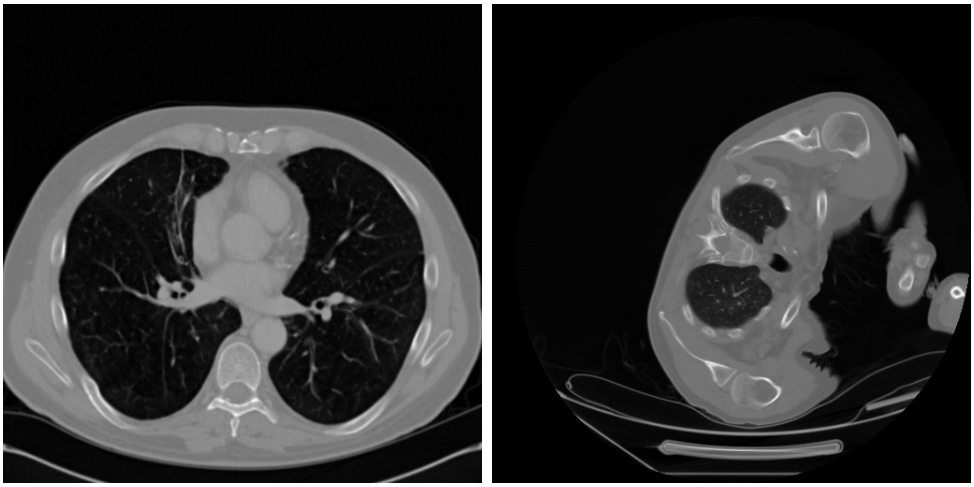
Di seguito, verranno effettuate delle valutazioni che terranno conto sia delle metriche che di una valutazione soggettiva per discutere meglio degli artefatti notati.

All'interno del dataset, saranno approfondite due immagini per valutare il comportamento della rete più da vicino. La scelta si è basata secondo alcuni parametri:

- L'immagine con blur deve avere un numero alto di artefatti rispetto all'immagine originale, in modo tale da valutare come l'immagine della rete li elimina.
- Le metriche dell'immagine di blur non devono avere un valore alto, mentre quelle dell'immagine della rete devono averne uno migliore, perlomeno nel caso del kernel lineare, così da evitare delle immagini che siano già compromesse in partenza all'interno del dataset.

Considerando questi vincoli, sono state scelte le immagini seguenti (Figura 3.4).

Per evitare di discutere degli stessi artefatti e dare maggiore risalto alla ricostruzione, verrà presa in considerazione la Figura 3.4a con kernel lineare, mentre la Figura 3.4b con il kernel di spline.



(a) (b)

Figura 3.4: Immagini scelte per valutazione

Analizzando le immagini ottenute con il kernel lineare (Figura 3.4a), si evidenzia un netto miglioramento in molti dettagli dell'immagine, supportato anche dalle metriche ottenute. Infatti, i valori di PSNR e SSIM aumentano rispettivamente del 18,64% e del 5,85%, raggiungendo i valori di 32,91 per la prima metrica e di 0,9 per la seconda, in piena linea con il valore mediano del *box-plot* di Figura 3.2.

Risultati ancora più soddisfacenti si ottengono con l'immagine con rumore, dimostrando come, con il kernel lineare, la rete si comporta in maniera eccellente sia con che senza rumore.

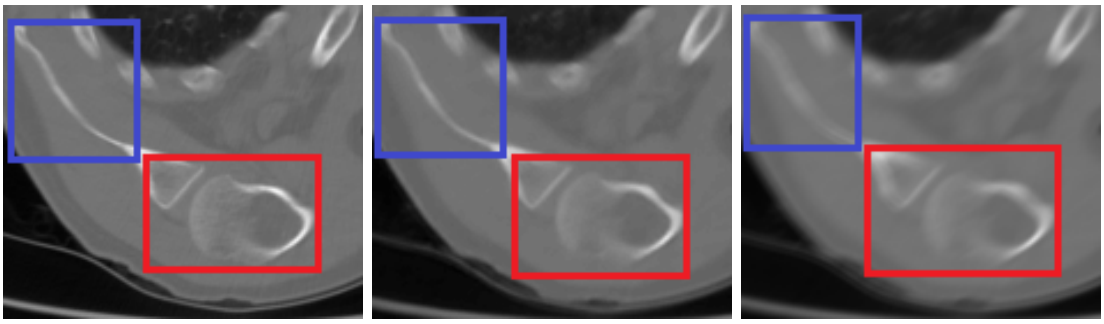
Scendendo nei dettagli, dalla Figura 3.5 si possono vedere dei particolari che differiscono da immagine a immagine. Per esempio, nel rettangolo blu della Figura 3.5c si perdono i dettagli soprattutto lungo i bordi della sezione presa in esame, non riuscendo a individuare a sufficienza la sua forma. Dall'altra parte, in Figura 3.5b, si mostra l'immagine ottenuta dalla rete che invece riproduce molto meglio questi dettagli. Gli stessi

(a) Originale (b) DeblurGAN-v2 (c) Blur

Figura 3.5: Confronto dettaglio delle immagini con kernel lineare di 3.4a

risultati si ottengono dalla zona individuata dal rettangolo rosso, in cui l'immagine di blur mostra un artefatto non riconducibile in nessun modo all'immagine originale.

Passando, invece, al kernel di spline applicato alla Figura 3.4b, si ottengono dei risultati contrastanti delle metriche. Mentre, tenendo in considerazione gli artefatti come nell'esempio precedente, i risultati sono promettenti. Infatti, i valori di PSNR e SSIM diminuiscono dall'immagine con blur a quella della rete, per esempio la seconda metrica nell'immagine senza rumore scende da 0,846 a 0,827, in piena linea con quanto mostrato nella Figura 3.3a.



(a) Originale (b) DeblurGAN-v2 (c) Blur

Figura 3.6: Confronto dettaglio delle immagini con kernel spline di 3.4b

Dal rettangolo blu (Figura 3.6c) si nota come l'intensità dei pixel sia diminuita, degradando in maniera netta l'informazione che invece si percepiva dalla Figura 3.6a. In

Figura 3.6b, invece, la ricostruzione ha ripristinato questo dettaglio.

Nel rettangolo rosso si può notare che sia l'immagine della rete che quella con la sfocatura introducono degli artefatti rispetto all'immagine originale. Infatti, risulta più difficile distinguere i bordi e la forma della sezione, che in queste tipologie di immagini è molto importante. Nonostante questo, la Figura 3.6b li delinea meglio, avvicinandosi all'immagine nitida (Figura 3.6a).

Una particolarità presente in tutta l'immagine, così come in altre del dataset, è la difficoltà nella rete a ricostruire le zone più grigie, questo, per esempio, si nota in Figura 3.6 sopra al rettangolo rosso. Infatti, sono molti i dettagli nell'immagine originale che evidenziano diverse forme nettamente mentre, nelle altre due, questa accuratezza si perde, rendendo questa parte meno precisa.

3.3 Confronto con SelfDeblur

Per l'ultima prova vengono confrontati i risultati ottenuti tra la rete esaminata in questo lavoro, DeblurGAN-v2, e la rete SelfDeblur.

SelfDeblur è composta da due reti generative che generano l'immagine non corrotta e il kernel di convoluzione. La prima adotta la tecnica di *Deep Image Prior* che utilizza reti neurali non allenate per ottenere statistiche di basso livello di un'immagine; la seconda, invece, è una rete totalmente connessa — *Fully-Connected Network* — che genera il kernel di sfocatura.

Per la ricostruzione è stata usata la Figura 3.4b, sia nel caso con rumore che senza. Inoltre, per l'immagine con rumore è stato adottato l'algoritmo di *Canny Edge Detection* nella rete SelfDeblur, in modo tale da ricavare un miglior risultato. Questo metodo verrà indicato come $SelfDeblur_{ce}$.

Metrica	$Blur$	$DeblurGAN-v2$	$SelfDeblur$
PSNR	26.169	25.827	**27.601**
SSIM	0.846	0.827	**0.880**

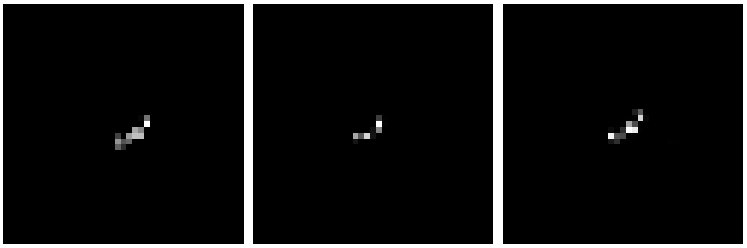
Tabella 3.1: Metriche Figura 3.4b senza rumore

Nelle Tabelle 3.1 e 3.2 si possono vedere i risultati ottenuti con le due reti a partire dall'immagine con *blur* di Figura 3.4b, sia senza che con il rumore. I valori delle metriche con SelfDeblur sono superiori rispetto all'immagine con sfocatura e, soprattutto, rispetto a DeblurGAN-v2. In particolare, si può notare come i risultati dell'immagine con rumore superino quelle senza rumore, infatti, come dimostrato in [14], con la rete di SelfDeblur con *edge detection* si ottengono dei risultati migliori se viene aggiunto del rumore gaussiano all'immagine con sfocatura. In Figura 3.8 si possono confrontare i risultati.

Metrica	$Blur$	$DeblurGAN-v2$	$SelfDeblur_{ce}$
PSNR	26.042	25.784	**28.841**
SSIM	0.790	0.782	**0.871**

Tabella 3.2: Metriche Figura 3.4b con rumore

Dato che la rete di SelfDeblur genera anche il kernel di convoluzione, in Figura 3.7 è stato fatto un confronto tra quello originale e i diversi risultati ottenuti con la rete. A conferma di quanto detto prima, il kernel di convoluzione è rappresentato in modo più accurato nel caso dell'immagine con rumore.

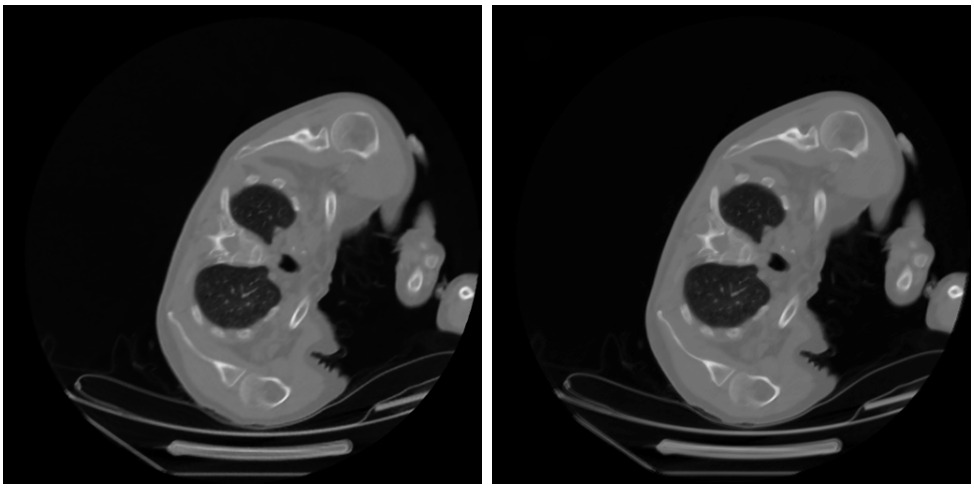


(a) Originale (b) Senza rumore (c) Con rumore

Figura 3.7: Confronto kernel di spline con SelfDeblur

Nonostante i risultati ottenuti con SelfDeblur siano superiori utilizzando un kernel più reale, il suo limite è nei tempi di computazione. La ricostruzione di un'immagine è esponenziale nella sua dimensione[14], quindi, per una delle immagini del dataset, di dimensione pari a 512×512 pixel, è necessario un tempo di 35 minuti. Dall'altra parte, la

rete DeblurGAN-v2 ha dei tempi nettamente inferiori: per una singola immagine impiega circa 1,70 secondi, rendendola molto più efficiente, nonostante i risultati ottenuti siano di poco inferiori.



(a) SelfDeblur (b) DeblurGAN-v2

Figura 3.8: Confronto Figura 3.4b tra SelfDeblur e DeblurGAN-v2 senza rumore

Conclusioni

In questo lavoro sono state prese in esame la convoluzione, il suo problema inverso e la difficoltà nella ricostruzione dell'immagine quando non si hanno informazioni sul filtro applicato. Sono state spiegate alcune principali applicazioni con un approfondimento della sfocatura da movimento, in particolare nelle sue tipologie e proprietà.

Sono state introdotte brevemente i tipi di rete neurale che trattano la rimozione della sfocatura, per poi focalizzarsi sulle reti generative avversarie e, in particolare, la rete neurale utilizzata in questa tesi, ovvero la DeblurGAN-v2.

Prendendo in considerazione un dataset di immagini tomografiche, sono stati analizzati i risultati ottenuti. Per questo mi sono avvalso di una serie di parametri da valutare per dare una panoramica il più possibile generale sul comportamento della rete. Per questo, sono stati considerati due tipologie di kernel di sfocatura, per rispecchiare sia casi più reali e sia più semplici, che casi con e senza rumore additivo.

In conclusione, la rete ha avuto degli ottimi risultati con il kernel lineare, mentre con il kernel di spline sono stati più negativi. Questo comportamento è giustificato dai dataset utilizzati durante l'allenamento della rete, che non rispecchiano invece le immagini mediche impiegate durante il lavoro.

Per questi motivi, se la rete in futuro venisse allenata con delle immagini più adatte, si otterrebbero dei risultati migliori e in linea con quelli ricavati utilizzando il filtro di sfocatura lineare.

Bibliografia

[1] H. Yang, X. Wu, and X. Sun, "Select Good Regions for Deblurring based on Convolutional Neural Networks," arXiv:2008.05065 [cs], Aug. 2020

[2] A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII, vol. 7578. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[3] X. Xu, J. Pan, Y.-J. Zhang, and M.-H. Yang, "Motion Blur Kernel Estimation via Deep Learning," IEEE Transactions on Image Processing, vol. 27, no. 1, pp. 194–205, Jan. 2018.

[4] Y. Li, Y. Luo, G. Zhang, and J. Lu, "Single image deblurring with cross-layer feature fusion and consecutive attention," Journal of Visual Communication and Image Representation, vol. 78, p. 103149, Jul. 2021.

[5] J. Koh, J. Lee, and S. Yoon, "Single-image deblurring with neural networks: A comparative survey," Computer Vision and Image Understanding, vol. 203, p. 103134, Feb. 2021.

[6] S. Nah, T. H. Kim, and K. M. Lee, "Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring," arXiv:1612.02177 [cs], May 2018.

[7] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv:1406.2661 [cs, stat], Jun. 2014.

[8] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang, "DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019, pp. 8877–8886.

[9] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, pp. 8183–8192.

[10] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," arXiv:1701.07875 [cs, stat], Dec. 2017.

[11] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep Video Deblurring for Hand-Held Cameras," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 237–246.

[12] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for Speed: A Benchmark for Higher Frame Rate Object Tracking," arXiv:1703.05884 [cs], Mar. 2017.

[13] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia, "Scale-Recurrent Network for Deep Image Deblurring," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, pp. 8174–8182.

[14] E. Marchidan, "Rimozione di artefatti da movimento in un'immagine con tecniche di deep learning", 2021.

[15] Albertina, B., Watson, M., Holback, C., Jarosz, R., Kirk, S., Lee, Y., Lemmerman, J., "Radiology Data from The Cancer Genome Atlas Lung Adenocarcinoma [TCGA-LUAD] collection. The Cancer Imaging Archive," 2016.

[16] Clark K, Vendt B, Smith K, Freymann J, Kirby J, Koppel P, Moore S, Phillips S, Maffitt D, Pringle M, Tarbox L, Prior F. The Cancer Imaging Archive (TCIA), "Maintaining and Operating a Public Information Repository, Journal of Digital Imaging," Volume 26, Number 6, December, 2013, pp 1045-1057.

[17] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[18] A. Chakrabarti, "A Neural Approach to Blind Motion Deblurring," arXiv:1603.04771 [cs], Aug. 2016.

Ringraziamenti

In questa ultima pagina, vorrei dedicare un pensiero a chi mi ha supportato durante il lavoro di questa tesi e in generale in questi tre magnifici anni.

Vorrei ringraziare la mia relatrice, la professoressa Elena Loli Piccolomini, per avermi proposto questo lavoro e per essere stata sempre disponibile per qualsiasi dubbio che potessi avere.

Un grazie va ai miei genitori, per avermi dato la possibilità di venire a Bologna per poter studiare quello che ho sempre voluto e per aver sempre creduto nelle mie possibilità, non potrei chiedere genitori migliori di voi.

Ringrazio la mia fidanzata Elena, per essere sempre stata vicina a me in questi anni e soprattutto durante questa tesi. Un grazie per avermi ascoltato e per avermi aiutato quando ero più scoraggiato, senza di te sarebbe stato tutto molto più difficile.

Infine, vorrei ringraziare i miei più grandi amici e coinquilini, Edoardo ed Elia. Grazie per questi anni di convivenza, per esserci aiutati a vicenda nei momenti più difficili e per tutti i momenti divertenti che abbiamo passato insieme.