Alma Mater Studiorum · Università di Bologna

Scuola di scienze Sede di Bologna
Dipartimento di Informatica - Scienza e Ingegneria
- DISI -

Modelli Diffusivi per l'Imaging Medico

Laurea Triennale in
Informatica

Relatore:
Chiar.mo Prof.
Davide Evangelista

Correlatore:
Chiar.mo Prof.
Elena Loli Piccolomini

Presentata da:
Luca Panariello

II Sessione
Anno Accademico 2023/2024

Al Fante di Coppe

Sommario

Questa tesi esplora l'impiego dei modelli diffusivi per la generazione e manipolazione di immagini mediche, con particolare attenzione ai processi di inversione e interpolazione.

Viene fornita una descrizione dettagliata del funzionamento dei modelli diffusivi e delle loro applicazioni nel contesto medico, evidenziando come possano essere utilizzati per generare immagini realistiche e correggere immagini corrotte o incomplete.

Il focus principale è posto sull'inversione dei modelli diffusivi, un processo volto a recuperare, data un'immagine il valore delle variabili delle quali l'immagine viene generata, e sulle sue potenziali applicazioni in ambito medico.

Le prestazioni degli esperimenti sono state valutate attraverso diverse metriche di qualità, dimostrando l'efficacia dei modelli nella ricostruzione accurata delle immagini originali.

Indice

1 Introduzione **3**

2 Modelli Diffusivi **4**
2.1 Denoising Diffusion Probabilistic Model 4
2.1.1 Forward Process 5
2.1.2 Backward Process 6
2.1.3 Training 7
2.1.4 Sampling 8
2.1.5 Prestazioni 9
2.2 Denoising Diffusion Implicit Model 10

3 Framework e Risultati **11**
3.1 Configurazioni 11
3.2 Hugging Face - Diffusers 12
3.3 Applicazioni: Denoising 14
3.3.1 Rumore Gaussiano 14
3.3.2 Rumore Speckle 15
3.3.3 Rumore Shot 15
3.3.4 Rumore con Scheduling Inverso 16
3.3.5 Confronto delle Metriche di Denoising 17

4 Inversione dei Modelli Diffusivi **18**
4.1 Inversione basata su Ottimizzazione 18
4.2 Inversione con Scheduler Inverso 20
4.2.1 Confronto Metriche 22
4.3 Applicazioni: Interpolazione 23

4.3.1 Interpolazione con Ottimizzazione 24
4.3.2 Interpolazione con Scheduler Inverso 25
4.3.3 Confronto Metodi di Interpolazione 26

5 Conclusioni **27**
5.1 Sviluppi Futuri 27
5.1.1 Inversione basata su Apprendimento 28
5.1.2 Manipolazione del Rumore 28

A Appendice **29**
A.1 Processi Stocastici e Catene di Markov 29
A.1.1 Processi Stocastici 29
A.1.2 Catene di Markov 29
A.2 Metriche 30
A.2.1 RMSE 30
A.2.2 PSNR 31
A.2.3 SSIM 31

Capitolo 1

Introduzione

Negli ultimi anni, i **modelli generativi a variabili latenti** hanno assunto un ruolo centrale nell'ambito dell'intelligenza artificiale, mostrando capacità straordinarie nella creazione di contenuti visivi. Questi modelli imparano a generare dati realistici a partire da una rappresentazione latente, una codifica astratta e ridotta dei dati originali, che ne cattura le caratteristiche fondamentali.

Tra i vari approcci generativi, i **modelli diffusivi** si sono distinti per la loro efficacia nella produzione di immagini realistiche. Questi modelli operano attraverso un processo iterativo che prevede l'aggiunta e la successiva rimozione di rumore, simulando il degrado e la ricostruzione dell'immagine in maniera controllata.

Inizialmente introdotti per la sintesi di immagini, i modelli diffusivi stanno trovando applicazioni sempre più rilevanti in settori come la medicina. Le immagini mediche, infatti, sono essenziali per la diagnosi e il monitoraggio di molte malattie, ma spesso soffrono di problemi di qualità, come la presenza di rumore o artefatti. L'elaborazione avanzata di queste immagini è quindi cruciale per migliorare la precisione diagnostica.

I modelli diffusivi offrono un approccio innovativo alla generazione di immagini sintetiche e alla manipolazione di immagini esistenti, permettendo non solo di migliorare la qualità visiva delle immagini, ma anche di recuperare informazioni da dati corrotti. Questo può avere un impatto significativo nell'analisi automatica delle immagini mediche e nella ricerca clinica.

In questa tesi, verranno esaminati i principi alla base dei modelli diffusivi, con particolare attenzione alle loro applicazioni nel settore medico. Saranno discussi in dettaglio i processi di rimozione del rumore e di inversione, due tecniche fondamentali che consentono di ricostruire immagini compromesse e di generare immagini sintetiche fedeli alla realtà.

Infine, attraverso un'analisi sperimentale, verranno confrontati i risultati ottenuti, valutando l'efficacia delle tecniche proposte utilizzando diverse metriche di qualità.

Capitolo 2

Modelli Diffusivi

I modelli diffusivi sono una categoria di modelli generativi a variabili latenti, progettati per apprendere a generare dati attraverso un processo di diffusione, che consiste in una catena di Markov di variabili aleatorie assolutamente continue. Vedere appendice A.1 per ulteriori dettagli.

Un modello diffusivo è caratterizzato da due componenti principali:

- Un processo di diffusione in avanti (*forward process*), che descrive come il segnale viene degradato progressivamente in rumore.
- Un processo inverso (*backward process*), che descrive come il segnale viene ricostruito, partendo da un rumore puro, tramite l'ausilio di una rete neurale.

In questo documento saranno presentati e analizzati due tipi di modelli diffusivi:

- **Denoising Diffusion Probabilistic Model (DDPM 2.1)**: un modello diffusivo proposto da Ho et al. [1], il cui processo inverso è stocastico.
- **Denoising Diffusion Implicit Model (DDIM 2.2)**: un modello diffusivo proposto da Song et al. [2], basato su DDPM ma con un processo inverso deterministico.

2.1 Denoising Diffusion Probabilistic Model

Dato un campione non perturbato $\mathbf{x}_0$, è possibile degradarlo progressivamente aggiungendo rumore gaussiano $\boldsymbol{\epsilon}^{(t)}$ a ogni passo temporale $t \in \{1, \dots, T\}$. L'intensità del rumore è determinata da una successione di parametri $\beta_{t\in\{1,\dots,T\}} \subseteq]0, 1[$, che definiscono lo *schedule di varianza*. In particolare, $\mathbf{x}_t$ viene calcolato come segue:

$$\mathbf{x}_t = \sqrt{\beta_t}\mathbf{x}_{t-1} + \sqrt{1-\beta_t}\boldsymbol{\epsilon}^{(t-1)}, \quad \text{dove} \quad \boldsymbol{\epsilon}^{(t-1)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.1}$$

per $T \ll 1$, si ottiene un campione $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

2.1.1 Forward Process

Il processo di diffusione in avanti può essere modellato come una catena di Markov con probabilità di transizione q. In particolare, l'aggiunta di rumore al campione $\mathbf{x}_{t-1}$ per ottenere $\mathbf{x}_t$ è descritta dalla relazione:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) \sim \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2.2}$$

In altre parole, il rumore viene campionato da una distribuzione normale multivariata con media $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$ e varianza $\beta_t\mathbf{I}$. Grazie alla proprietà markoviana della catena A.1.1, è possibile descrivere l'intero processo di diffusione in avanti come:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad \text{con} \quad \mathbf{x}_{1:T} = \mathbf{x}_1, \ldots, \mathbf{x}_T \tag{2.3}$$

Infine, è possibile campionare direttamente $\mathbf{x}_t$ da $\mathbf{x}_0$ utilizzando la riparametrizzazione. Definendo $\alpha_t = 1 - \beta_t$ e $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, si ottiene il seguente risultato:

Osservazione 2.1.1: Trucco della Riparametrizzazione

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}^{(t-1)} \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^{(t-1)} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}^{(t-2)}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^{(t-1)} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}^{(t-2)} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}^{(t-1)} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t}\boldsymbol{\epsilon}_{t-2} \\
&\overset{1}{=} \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1 - \alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} \\
&= \ldots \\
&= \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}_t.
\end{aligned}$$

Dove $\boldsymbol{\epsilon}^{(i)}$ rappresenta il rumore generato al passo i, mentre $\boldsymbol{\epsilon}_i$ indica il rumore cumulato fino al passo i.

[1] La somma di variabili gaussiane indipendenti è ancora una variabile gaussiana, con media pari alla somma delle medie e varianza pari alla somma delle varianze.

Pertanto, $\mathbf{x}_t$ può essere campionato da una distribuzione normale con media $\sqrt{\overline{\alpha}_t}\mathbf{x}_0$ e varianza $(1-\overline{\alpha}_t)\mathbf{I}$, in un solo passo:

$$q(\mathbf{x}_t|\mathbf{x}_0) \sim \mathcal{N}(\sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}) \tag{2.4}$$

2.1.2 Backward Process

Il processo inverso, a differenza di quello di diffusione in avanti, ha l'obiettivo di ricostruire il campione originale $\mathbf{x}_0$ a partire dal campione rumoroso $\mathbf{x}_T$.

La costruzione esplicità del processo inverso è più complessa, poiché il calcolo diretto della distribuzione condizionale $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ richiederebbe l'uso dell'intero dataset di addestramento.

Infatti, applicando il Teorema di Bayes a $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, si ha

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})q(\mathbf{x}_{t-1})}{q(\mathbf{x}_t)} = q(\mathbf{x}_t|\mathbf{x}_{t-1})\frac{\sum\limits_{\mathbf{x}_0 \in \text{Dataset}} q(\mathbf{x}_{1:t-1}|\mathbf{x}_0)}{\sum\limits_{\mathbf{x}_0 \in Dataset} q(\mathbf{x}_{1:t}|\mathbf{x}_0)} \tag{2.5}$$

Tale complessità può essere risolta utilizzando una Rete Neurale Profonda (Deep Neural Network, DNN), costruita per risolvere il task di Denoising, come una U-Net[3] o un Transformer [4], per predire il rumore da rimuovere.

Per semplicità, indicheremo indicato con $\boldsymbol{\varepsilon_\theta}$.

Durante la fase di addestramento, andremo quindi ad ottimizzare i parametri $\boldsymbol{\theta}$ della DNN per costruire una distribuzione $p_{\boldsymbol{\theta}}$ che approssima il processo inverso $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, che elimina progressivamente il rumore da $\mathbf{x}_T$ fino a ricostruire il dato originale.

Nei DDPM, il processo inverso è stocastico e modellato come una catena di Markov con transizione definita dalla seguente espressione:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)) \tag{2.6}$$

Dove $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ e $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ rappresentano rispettivamente la media e la varianza del rumore predetto dalla DNN, che prende come input il campione $\mathbf{x}_t$ e il tempo t.

Grazie alla natura markoviana della catena A.1.1, è possibile decomporre l'intero processo inverso come segue:

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{0:t-1}|\mathbf{x}_T) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.7}$$

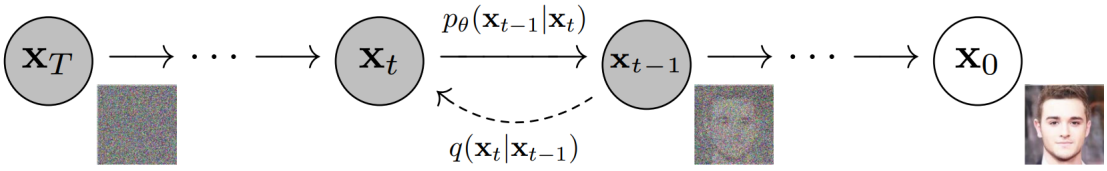


Figura 2.1: Processi Forward e Backward di un DDPM (figura da Ho. et al [1])

2.1.3 Training

Nonostante il processo q non sia facilmente calcolabile, è comunque possibile stimare una transizione condizionata al campione originale $\mathbf{x}_0$. Questo è realizzabile attraverso la regola di Bayes e alcune proprietà della probabilità condizionata.

In particolare, si ha:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_{0,t}) &= \frac{q(\mathbf{x}_{t-1,t,0})}{q(\mathbf{x}_{t,0})} \\ &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1,0})q(\mathbf{x}_{t-1,0})}{q(\mathbf{x}_t|\mathbf{x}_0)q(\mathbf{x}_0)} \\ &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1,0})q(\mathbf{x}_{t-1}|\mathbf{x}_0)q(\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)q(\mathbf{x}_0)} \\ &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1,0})q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \end{aligned}$$

Attraverso questa metodologia, possiamo ricavare la media e la varianza corrette del processo inverso $q(\mathbf{x}_{t-1}|\mathbf{x}_{t,0})$ ovvero:

$$\boldsymbol{\mu}_t = \frac{1}{\sqrt{\overline{\alpha}_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_t\right) \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.8}$$

$$\boldsymbol{\Sigma}_t = \sigma_t^2\mathbf{I} = \left(\frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\cdot\beta_t\right)\mathbf{I} \tag{2.9}$$

Osserviamo che la varianza è funzione esclusivamente dello schedule di varianza, il che permette di ottimizzare i parametri in funzione della sola media e calcolare una funzione di loss.

$$\mathcal{L}_t(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\boldsymbol{\mu}_t - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right\|^2\right] + C \tag{2.10}$$

Grazie all'uso di una rete neurale di denoising (DNN), è possibile predire la media e la varianza del rumore, consentendo di calcolare la funzione di loss.

$$\begin{aligned}
\mathcal{L}_t(\boldsymbol{\theta}) &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\overline{\alpha}_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\epsilon}_t\right) - \frac{1}{\sqrt{\overline{\alpha}_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right)\right\|^2\right] + C \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|-\frac{\beta_t}{\sqrt{\overline{\alpha}_t(1-\overline{\alpha}_t)}}\boldsymbol{\epsilon}_t + \frac{\beta_t}{\sqrt{\overline{\alpha}_t(1-\overline{\alpha}_t)}}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\right\|^2\right] + C \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\cdot\overline{\alpha}_t(1-\overline{\alpha}_t)}\left\|\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \boldsymbol{\epsilon}_t\right\|^2\right] + C
\end{aligned}$$

Semplificando ulteriormente la funzione di loss, come proposto da Ho et al. [1], ed eliminando alcuni parametri costanti legati alla varianza, si riduce il rumore appreso dalla rete. Inoltre, l'introduzione di un campionamento del passo temporale t da una distribuzione uniforme rende l'ottimizzazione dei parametri della rete più efficiente, favorendo una migliore convergenza del modello e migliorando la qualità dei risultati.

$$L_t(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[\left\|\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \boldsymbol{\epsilon}_t\right\|^2\right] \quad t \sim \text{Uniform}(\{1,\ldots,T\}) \tag{2.11}$$

In definitiva, i parametri $\boldsymbol{\theta}$ della DNN vengono ottimizzati iterativamente attraverso un processo di apprendimento descritto nell'Algoritmo 1:

Algoritmo 1 Training di DDPM

loop
  $\mathbf{x}_0 \sim$ Dataset
  $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
  $t \sim \text{Uniform}(\{1,\ldots,T\})$
  $\mathbf{x}_t = \sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}_t$
  $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}}\left[\left\|\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \boldsymbol{\epsilon}_t\right\|^2\right]$
  if `Converge` **then**
    return $\boldsymbol{\theta}^*$
  end if
end loop

2.1.4 Sampling

Una volta addestrato il modello, i parametri appresi possono essere utilizzati per generare nuovi campioni. A partire da un campione rumoroso $\mathbf{x}_t$, è possibile calcolare $\mathbf{x}_{t-1}$

utilizzando la relazione (2.6), che combina la media $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ e la varianza $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ del rumore predetto dalla rete.

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) + \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\boldsymbol{\epsilon}_t \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.12}$$

A partire dal campione rumoroso iniziale $\mathbf{x}_T$, la procedura utilizza la parametrizzazione della media e della varianza, come descritto nelle equazioni (2.8) e (2.9), e viene ripetuta attraverso una serie di passaggi iterativi fino a ottenere il campione finale $\mathbf{x}_0$, che rappresenta un'immagine generata dal modello.

L'Algoritmo 2 descrive il processo di campionamento per DDPM:

Algoritmo 2 Sampling di DDPM

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T$ **to** 1 **do**
3: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$ else $\boldsymbol{\epsilon} = \mathbf{0}$
4: $\quad \mu_t = \frac{1}{\sqrt{\overline{\alpha}_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}} \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \right)$
5: $\quad \sigma_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t} \cdot \beta_t$
6: $\quad \mathbf{x}_{t-1} = \mu_t + \sigma_t \boldsymbol{\epsilon}$
7: **end for**
8: **return** $\mathbf{x}_0$

2.1.5 Prestazioni

Il modello DDPM è stato testato su diversi dataset, dimostrando di poter generare immagini di qualità comparabile a quelle delle GANs, ma con maggiore stabilità e semplicità nel processo di addestramento. Tuttavia, presenta alcune limitazioni, tra cui:

- **Lentezza computazionale**: Per ottenere campioni finali di alta qualità, è spesso necessario impostare un numero elevato di passi, tipicamente $T = 1000$, il che rende il processo di generazione piuttosto lento.
- **Dipendenza dallo schedule di varianza**: La qualità dei campioni è sensibile alla scelta dello schedule di varianza. Un'errata configurazione può portare a risultati di scarsa qualità. Solitamente, si sceglie uno schedule con $\beta_{t-1} < \beta_t$ che cresce in modo lineare.

2.2 Denoising Diffusion Implicit Model

Il modello DDIM, proposto da Song et al. [2], è una variante del DDPM che affronta uno dei principali limiti di quest'ultimo: l'uso di molti passi temporali per ottenere campioni di alta qualità.

Il DDIM si basa sul DDPM, ma introduce un processo inverso deterministico. Come evidenziato nell'osservazione 2.1.1, è possibile ottenere un campione $\mathbf{x}_{t-1}$ scomponendo i rumori gaussiani:

$$\begin{aligned}\mathbf{x}_{t-1} &= \sqrt{\overline{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_{t-1}}\boldsymbol{\epsilon}_{t-1} \\ &= \sqrt{\overline{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2+\sigma_t^2}\boldsymbol{\epsilon}_{t-1} \\ &= \sqrt{\overline{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2}\boldsymbol{\epsilon}_t + \sqrt{\sigma_t^2}\boldsymbol{\epsilon}^{(t-1)}\end{aligned}$$

Usando la DNN per predire il rumore $\boldsymbol{\epsilon}_t$ e stimare il campione $\mathbf{x}_0$, come descritto nell'osservazione 2.1.1, è possibile ottenere il campione $\mathbf{x}_{t-1}$ direttamente da $\mathbf{x}_t$:

$$\mathbf{x}_{t-1} = \sqrt{\overline{\alpha}_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sqrt{\overline{\alpha}_t}}\right) + \sqrt{1-\overline{\alpha}_{t-1}-\sigma_t^2}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) + \sqrt{\sigma_t^2}\boldsymbol{\epsilon}^{(t-1)} \quad (2.13)$$

La varianza σ_t^2 è ciò che introduce la casualità nel processo. Tuttavia, ponendo $\sigma_t^2 = 0$, si ottiene un processo inverso completamente deterministico.

Inoltre, è possibile calcolare un campione $\mathbf{x}_s$ con $s < t$, saltando alcuni passi temporali tramite la transizione $q(\mathbf{x}_s|\mathbf{x}_{t\cap 0})$, ottenendo:

$$\mathbf{x}_s = \sqrt{\overline{\alpha}_s}\left(\frac{\mathbf{x}_t - \sqrt{1-\overline{\alpha}_t}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)}{\sqrt{\overline{\alpha}_t}}\right) + \sqrt{1-\overline{\alpha}_s}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) \quad (2.14)$$

Grazie a questo approccio, il modello DDIM è in grado di generare campioni di alta qualità utilizzando un numero ridotto di passi temporali. Song et al. [2] hanno dimostrato che sono sufficienti $S = 100$ passi, riducendo significativamente il tempo di generazione delle immagini.

Capitolo 3

Framework e Risultati

3.1 Configurazioni

La totalità degli esperimenti è stato svolto in linguaggio Python, e usando principalmente le librerie PyTorch e Diffusers.

Ad eccezione del training dei modelli diffusivi, gli esperimenti sono stati eseguiti su un personal computer avente le seguenti specifiche:

- **CPU**: Intel Core(R) i7-1280P (14 cores, 20 threads, 4.80 GHz)
- **GPU**: NVIDIA GeForce GTX 1650 with Max-Q Design (4GB)
- **CUDA**: 12.5
- **Python**: 3.12.4
- **PyTorch**: 2.3.1+cu121
- **Diffusers**: 0.29.2

Come descritto nell'articolo di Bolón-Canedo et al. [5], l'utilizzo di un computer personale, favorisce una gestione più controllata delle risorse computazionali. Questo approccio, in linea con i principi del Green AI, contribuisce a ridurre l'impatto ambientale ottimizzando il consumo energetico rispetto all'uso di grandi infrastrutture cloud. Inoltre, l'uso di risorse locali consente di migliorare l'efficienza energetica, riducendo così l'impronta di carbonio associata all'esecuzione delle analisi.

3.2 Hugging Face - Diffusers

Gli esperimenti e i risultati di questa tesi sono stati ottenuti utilizzando la libreria *Diffusers*, parte del framework *Hugging Face*. Diffusers offre tre strutture fondamentali:

- **Modelli**: reti neurali che stimano il rumore $\boldsymbol{\epsilon}$ (output) partendo da un'immagine $\mathbf{x}$ a uno step temporale t dato in input. $[\boldsymbol{\epsilon} = M(\mathbf{x}, t)]$
- **Scheduler**: implementa il processo di diffusione aggiungendo o rimuovendo il rumore stimato dal modello, in base al tipo di scheduler utilizzato. $[\mathbf{x} = S(\boldsymbol{\epsilon}, \mathbf{x}, t)]$
- **Pipeline**: combina modello e scheduler per generare nuove immagini, attraverso il processo di diffusione.

In particolare, è stato analizzato lo scheduler DDIM, utilizzando il dataset Mayo [6], composto da radiografie toraciche. Le immagini sono state ridimensionate a 128×128 pixel e normalizzate su un unico canale di grigi, con valori tra 0 e 1.

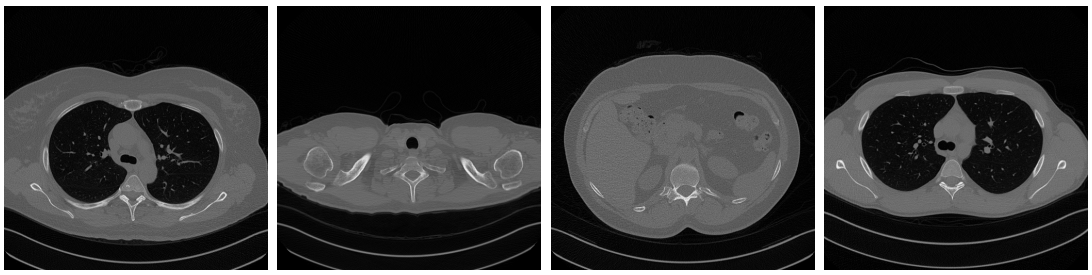


Figura 3.1: Immagini del dataset MAYO, usate per l'allenamento del modello

Il modello impiegato è una U-Net, organizzata in 4 blocchi di codifica e 4 di decodifica. I blocchi seguono questo ordine e sono simmetrici per ridurre e aumentare la dimensionalità:

- Blocco Convolutivo 2D
- Blocco Convolutivo 2D
- Blocco di Attenzione
- Blocco Convolutivo 2D

Ogni blocco è composto da due strati che generano tensori con un numero di canali crescente (64, 128, 256, 512) man mano che si scende in profondità.

Sono state allenate due reti neurali, una per ciascun modello diffusivo, seguendo l'Algoritmo 1. Durante l'addestramento, lo scheduler è stato impostato con $T = 1000$ passi di diffusione, mentre la varianza è stata regolata da $\beta_0 = 1e-4$ e $\beta_T = 1e-2$, con crescita lineare.

In Figura 3.2 sono riportate le immagini generate attraverso la pipeline:

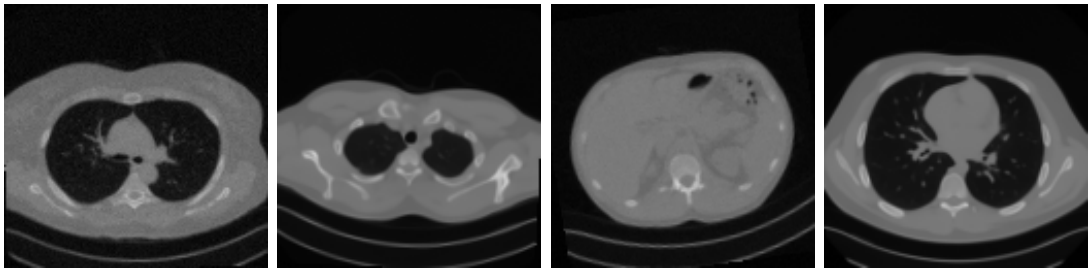


Figura 3.2: Immagini generate attraverso la pipeline

Come si può osservare, le immagini generate sono di ottima qualità e alcune di esse richiamano chiaramente quelle utilizzate durante il processo di addestramento, come mostrato in Figura 3.1. La lieve rotazione osservata è dovuto a una piccola variazione casuale introdotta nelle immagini di training durante l'apprendimento, con l'obiettivo di ridurre l'overfitting e simulare in modo più realistico la variabilità nella posizione del paziente durante una radiografia.

Infine, per gli esperimenti descritti di seguito, è stata implementata una funzione di sampling che prende in input un'immagine $\mathbf{x}$ e applica il processo di campionamento utilizzando un modello U-Net M per la rilevazione del rumore e uno scheduler DDIM S per la sua rimozione.

Algoritmo 3 Funzione di Sampling

function sampling($\mathbf{x}$,M,S,T):
  $\mathbf{x}_t = \mathbf{x}$
  for $t = T$ **to** 1:
    $\boldsymbol{\epsilon}_t = M(\mathbf{x}_t, t)$
    $\mathbf{x}_{t-1} = S(\boldsymbol{\epsilon}_t, \mathbf{x}_t, t)$
  end for
  return $\mathbf{x}_0$

3.3 Applicazioni: Denoising

I modelli diffusivi possono essere utilizzati anche per il *denoising*, ovvero la rimozione del rumore da un'immagine. Il processo di denoising può essere eseguito utilizzando la funzione di sampling, con un numero di passi di diffusione pari a T.

Fissato $T = 50$, e data un'immagine pulita $\mathbf{y}$, questa è stata prima corrotta presenti in seguito, per ottenere un'immagine corrotta $\mathbf{x}$, e successivamente è stato applicato il processo di backward descritto dall'Algoritmo 3 su $\mathbf{x}$ con lo scopo di ricostruire l'immagine originale su diverse tipologie di rumore descritte nell'articolo di Prasad et al. [7], ottenendo risultati descritti in seguito. Le metriche utilizzate per valutare la qualità delle immagini sono state RMSE, PSNR e SSIM (per le specifiche delle metriche si fa rifornimento alla Sezione A.2).

3.3.1 Rumore Gaussiano

Per corrompere un'immagine con rumore gaussiano, ho utilizzato la seguente espressione:

$$\mathbf{x} = \mathbf{y} + \sigma\boldsymbol{\epsilon} \quad \text{con} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{3.1}$$

dove σ è la deviazione standard, che controlla l'intensità del rumore.

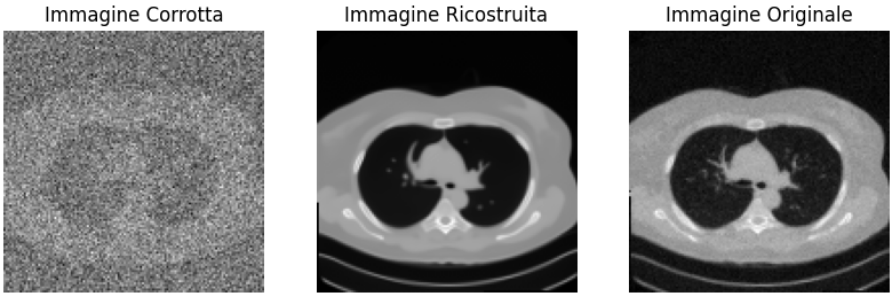


Figura 3.3: Rumore Gaussiano $\sigma = 1$

La generazione ha mostrato buone performance anche con rumore elevato, grazie all'allenamento mirato a riconoscere rumore gaussiano. Tuttavia, per $\sigma > 1$, il degrado dell'immagine è tale che la funzione non riesce a ripristinare efficacemente l'immagine originale.

3.3.2 Rumore Speckle

Il rumore speckle è un rumore moltiplicativo, tipico di immagini acquisite da sensori radar o sonar, come le radiografie. Può essere simulato con la formula:

$$\mathbf{x} = \mathbf{y} + \mathbf{y}\sigma\boldsymbol{\epsilon} \quad \text{con} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I}), \tag{3.2}$$

anche in questo caso, con un'intensità di rumore $\sigma \leq 2$, i risultati sono stati soddisfacenti:

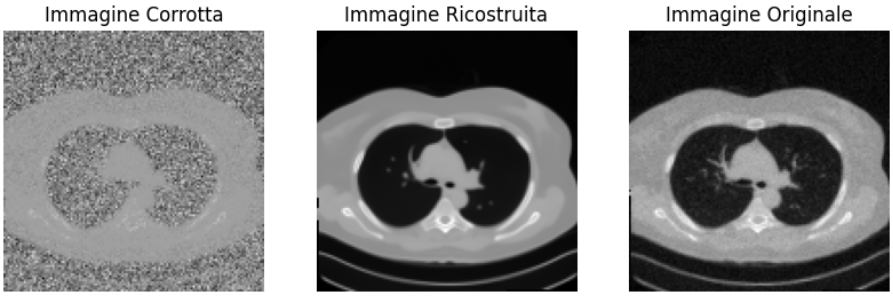


Figura 3.4: Rumore Speckle $\sigma = 2$

3.3.3 Rumore Shot

Il rumore shot è un tipo di rumore additivo che si manifesta in immagini acquisite da sensori ottici, come fotocamere digitali o telescopi, in condizioni di bassa illuminazione. L'immagine corrotta da rumore shot si ottiene come segue:

$$\mathbf{x} = \mathbf{y} + \boldsymbol{\epsilon} \quad \text{con} \quad \boldsymbol{\epsilon} \sim \text{Poisson}(\lambda), \tag{3.3}$$

dove λ è il parametro di intensità del rumore.

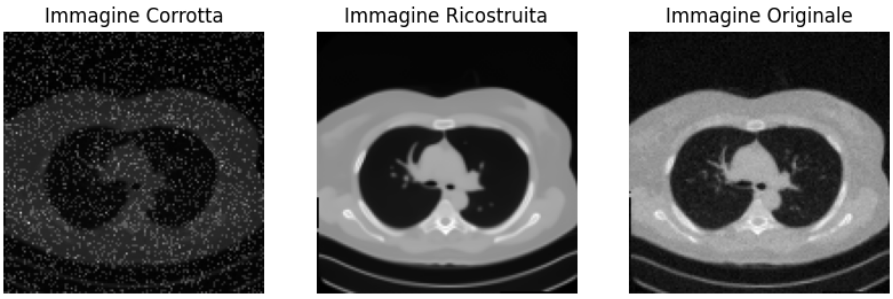


Figura 3.5: Rumore Shot $\lambda = 0.25$

Con $\lambda \geq 0.25$, l'immagine si oscura notevolmente e il denoising diventa inefficace, poiché il rumore Poisson non è stato considerato nell'allenamento del modello.

3.3.4 Rumore con Scheduling Inverso

Un'altra tipologia di rumore che può essere testata è quella con lo scheduling inverso. Infatti, dato uno scheduler DDIM S, è possibile creare uno scheduler inverso S^{-1}, il quale aggiunge rumore invece di rimuoverlo, mantenendo la stessa intensità basata sullo scheduler di varianza β_t. Il processo parte dal passo 0 e prosegue fino al passo T.

È quindi possibile implementare una funzione di "*noising*" che prende in input un'immagine pulita $\mathbf{y}$ e vi aggiunge rumore attraverso T passi, come descritto nell'Algoritmo 4:

Algoritmo 4 Funzione di Noising

function noising($\mathbf{y}$,M,S^{-1},T):
  $\mathbf{x}_0 = \mathbf{y}$
  for $t = 0$ **to** T:
    $\boldsymbol{\epsilon}_t = M(\mathbf{x}_t, t)$
    $\mathbf{x}_{t+1} = S^{-1}(\boldsymbol{\epsilon}_t, \mathbf{x}_t, t)$
  end for
  return $\mathbf{x}_T$

In Figura 3.6 si può notare come la ricostruzione sia praticamente identica all'immagine originale, poiché il rumore aggiunto e successivamente rimosso è lo stesso, grazie alla natura deterministica dello scheduler DDIM.

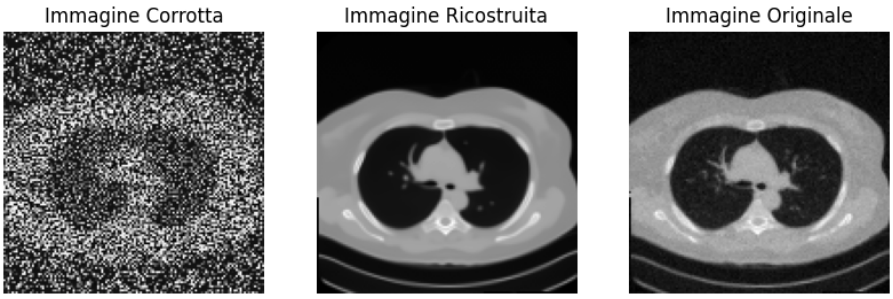


Figura 3.6: Rumore con Scheduling Inverso - $T = 30$

3.3.5 Confronto delle Metriche di Denoising

Di seguito sono riportate le metriche di qualità delle immagini ottenute con i diversi tipi di rumore e parametri testati:

Rumore	RMSE	PSNR	SSIM
Gaussiano con $\sigma = 0.5$	0.0536	25.4158	0.7563
Gaussiano con $\sigma = 1$	0.0544	25.2838	0.7288
Gaussiano con $\sigma = 2$	0.0833	21.5819	0.6654
Speckle $\sigma = 1$	0.0512	25.8180	0.7451
Speckle $\sigma = 2$	0.0641	23.8569	0.7124
Speckle $\sigma = 3$	0.0722	22.8307	0.7134
Shot $\lambda = 0.25$	0.0483	26.3138	0.7660
Shot $\lambda = 0.5$	0.0575	24.8022	0.7640
Shot $\lambda = 0.75$	0.0781	22.1445	0.7162
Scheduler - 10 timestep	0.0508	25.8857	0.7551
Scheduler - 20 timestep	0.0509	25.8579	0.7362
Scheduler - 30 timestep	0.0547	25.2435	0.7449

Tabella 3.1: Metriche di denoising con DDIM

Dalla Tabella 3.1 si osserva una rapida decrescita della qualità delle immagini all'aumentare dei valori dei parametri, in particolare per il rumore shot, mentre il rumore generato con lo scheduler tende a produrre risultati meno variabili rispetto al numero di passi di diffusione. È importante notare che, mentre rumori come il gaussiano, lo speckle e lo shot trovano applicazione in contesti reali, lo stesso non si può dire per il rumore generato dallo scheduler, che, come verrà mostrato nel capitolo successivo, ha altre applicazioni specifiche.

Capitolo 4

Inversione dei Modelli Diffusivi

In questo capitolo affrontiamo il problema dell'inversione dei modelli diffusivi, ovvero la generazione del rumore causale a partire da un'immagine. Formalmente, possiamo considerare un modello diffusivo già addestrato come una funzione D che prende in input del rumore casuale $\boldsymbol{\epsilon}$ e genera un'immagine $\mathbf{y}$. L'obiettivo dell'inversione è calcolare il processo inverso, ovvero la funzione D^{-1}.

Gli esperimenti e i risultati seguenti sono stati condotti su un'immagine generata dal modello (Figura 3.2) al fine di selezionare efficacemente i parametri da utilizzare durante l'inversione, sfruttando il processo generativo del modello.

4.1 Inversione basata su Ottimizzazione

Uno dei metodi per risolvere il problema dell'inversione è l'approccio basato sull'ottimizzazione, come descritto da Xia et al. [8].

Questo metodo minimizza una funzione di loss L:

$$\arg\min_{\boldsymbol{\epsilon}\in\mathbb{R}^n} L(D(\boldsymbol{\epsilon}), \mathbf{y}, \boldsymbol{\epsilon}) \tag{4.1}$$

La scelta dell'ottimizzatore è fondamentale; in questo caso, è stato utilizzato AdamW [9] con un learning rate fissato a 10^{-2} per garantire una rapida convergenza.

Anziché usare una funzione di loss standard, è stata utilizzata una funzione di loss personalizzata che tiene conto del rumore da ottimizzare e della regolarizzazione:

$$L(\mathbf{x}, \mathbf{y}, \boldsymbol{\epsilon}) = \sum_{i=1}^{n} (x_i - y_i)^2 + \lambda \sum_{i=1}^{n} \epsilon_i^2 \tag{4.2}$$

Dove $\boldsymbol{\epsilon}$ rappresenta il rumore da ottimizzare e λ è il parametro di regolarizzazione.

Dato che l'inversione è un problema mal posto, è necessario utilizzare una regolarizzazione per evitare risultati instabili, sono state testate diversi valori di λ per trovare il valore ottimale come descritto in Tabella 4.1.

In questo caso è stato scelto $\lambda = 10^{-1}$ considerando maggiormente il PSNR.

Parametro	RMSE	PSNR	SSIM
$\lambda = 10^{-4}$	0.0272	31.2933	0.8633
$\lambda = 10^{-3}$	0.0270	31.3742	0.8640
$\lambda = 10^{-2}$	0.0251	32.0057	0.8645
$\lambda = 10^{-1}$	0.0238	32.4747	0.8611
$\lambda = 1$	0.0735	22.6716	0.7471

Tabella 4.1: Metriche per diversi parametri di regolarizzazione

La procedura di generazione può essere realizzata utilizzando la funzione di sampling descritta nell'Algoritmo 3.

Fissando un numero massimo di iterazioni, un learning rate per l'ottimizzatore, un numero di passi di diffusione T, un DNN M e uno scheduler S, è possibile implementare una procedura di inversione basata su ottimizzazione che, partendo da un'immagine $\mathbf{y}$, restituisce il rumore $\boldsymbol{\epsilon}$ che la genera.

Algoritmo 5 Inversione basata su Ottimizzazione

function inversion_opt($\mathbf{y}$, T, max_iter):
  M = U-Net
  S = DDIM
  $\boldsymbol{\epsilon} = \mathcal{N}(0, 1)$
  for $i = 1$ **to** max_iter:
    $\mathbf{x}$ = sampling($\boldsymbol{\epsilon}, M, S, T$)
    $\boldsymbol{\epsilon}$ = AdamW($\boldsymbol{\epsilon}$, loss = $L(\mathbf{x}, \mathbf{y}, \boldsymbol{\epsilon})$, lr = $1e-2$)
  end for
  return $\boldsymbol{\epsilon}$

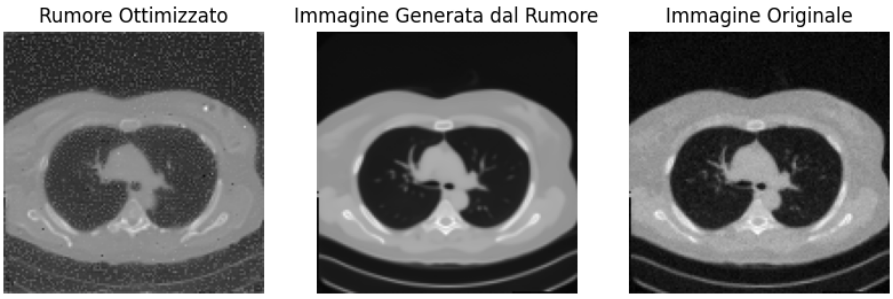


Figura 4.1: Risultati tramite Metodo di Ottimizzazione con $T = 10$ e $max\_it = 500$

Sebbene questo approccio produca risultati qualitativamente validi, esso richiede notevoli risorse computazionali e tempo. Per questo motivo, abbiamo esplorato un altro metodo che utilizza tecniche precedentemente discusse.

4.2 Inversione con Scheduler Inverso

Un'altra tecnica per invertire i modelli diffusivi consiste nell'utilizzare uno scheduler inverso. Data un'immagine pulita $\mathbf{y}$, è possibile applicare la procedura descritta nella sezione 3.3.4 per aggiungere rumore in modo graduale fino a ottenere un'immagine corrotta $\boldsymbol{\epsilon}$.

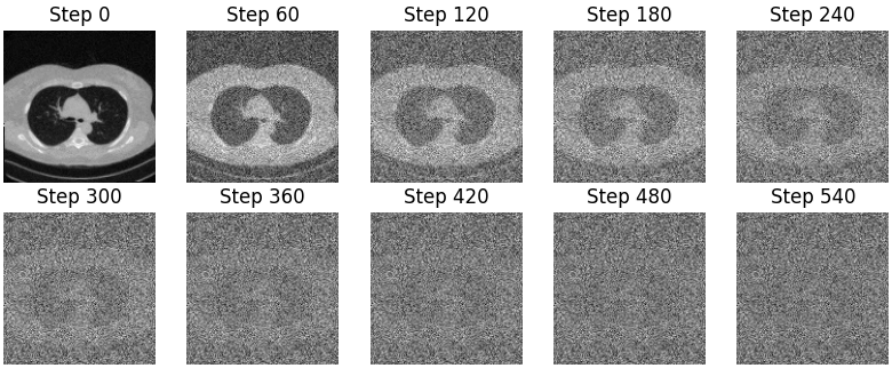


Figura 4.2: Applicazione della funzione di noising ogni 60 passi

La particolarità di questa tecnica è che il rumore viene aggiunto progressivamente e in maniera controllata. Successivamente, attraverso la funzione di denoising descritta nella sezione 3.3, è possibile rimuovere il rumore e recuperare l'immagine originale, rendendo così $\boldsymbol{\epsilon}$ il rumore che ha generato l'immagine $\mathbf{y}$.

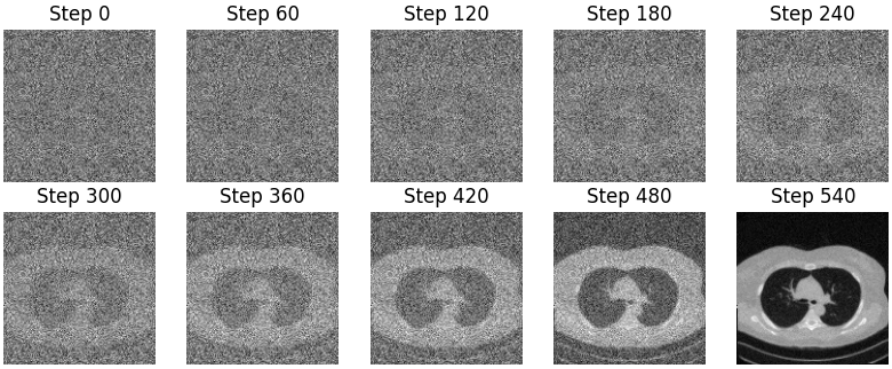


Figura 4.3: Applicazione della funzione di denoising ogni 60 passi

In un esperimento condotto con uno scheduler inverso DDIM, variando il numero di passi di diffusione T da 100 a 1000 con incrementi di 100, si è riscontrato che il valore ottimale di T è 600. Con questo valore, non solo le metriche di qualità risultano migliori, ma il test di normalità ha restituito un p-value maggiore, indicando che il rumore generato è più simile a un campione proveniente da una distribuzione normale.

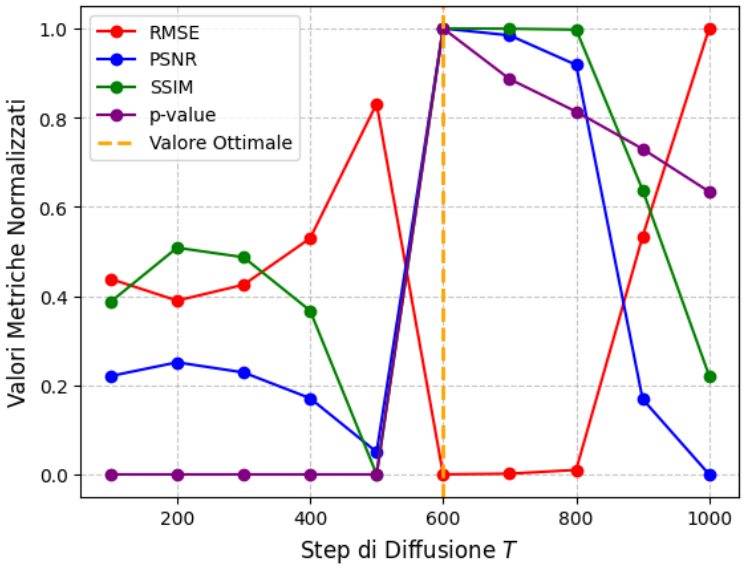


Figura 4.4: Grafico di andamento delle metriche al variare di T

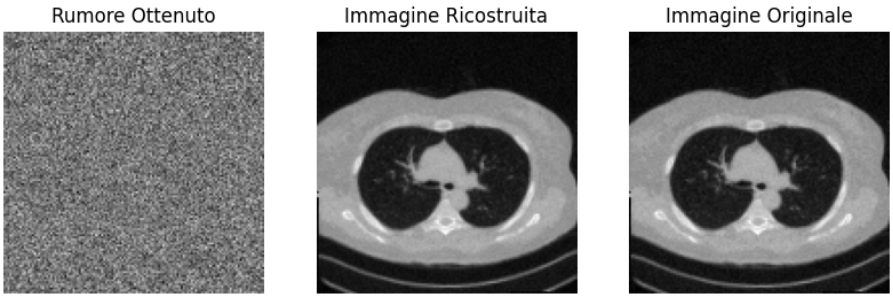


Figura 4.5: Risultato di Inversione con Immagine Generata

Ottimi risultati sono stati ottenuti anche con immagini test, non generate dal modello diffusivo, come la seguente:

Figura 4.6: Risultato di Inversione con Immagine di Test

In questo caso il rumore non assomiglia a quello generato da una distribuzione normale, questo perché l'aggiunta di rumore è determinata dalla DNN che predice il rumore da rilevare ad ogni passo, dale rete non è allenata per rilevare rumore da immagini che non riesce a generare. Nonostante ciò anche se il rumore non è normale, la ricostruzione dell'immagine è comunque valida; la DNN riesce a predirre ottimalmente il rumore da togliere ma non da aggiungere.

4.2.1 Confronto Metriche

Le metriche di qualità ottenute con i vari metodi di inversione sono riportate di seguito:

Inversione con Immagine	RMSE	PSNR	SSIM	p-value	Tempo
Generata - 200 passi	0.0542	25.3188	0.7972	0.0	24.9937
Generata - 400 passi	0.0723	22.8145	0.7397	0.0	50.7022
Generata - 600 passi	0.0037	48.612	0.9976	0.3456	79.4218
Generata - 800 passi	0.005	46.0684	0.9965	0.2811	106.714
Generata - 1000 passi	0.1333	17.5043	0.6803	0.2191	139.8648
Test - 200 passi	0.1462	16.6992	0.4641	0.0000	25.3172
Test - 400 passi	0.1135	18.8992	0.6306	0.0000	51.9980
Test - 600 passi	0.0170	35.4145	0.9814	0.0000	80.9647
Test - 800 passi	0.0311	30.1547	0.9512	0.0000	107.9655
Test - 1000 passi	0.1848	14.6680	0.4794	0.0000	135.7887

Tabella 4.2: Metriche di confronto tra diversi passi di diffusione T

Dalla Tabella 4.2 si osserva che i risultati migliori si ottengono a 600 passi. Superando questo numero, l'immagine corrotta diventa eccessivamente rumorosa, al punto che il rumore risulta così uniformato rendendo difficile la sua rilevazione da parte della rete. Al contrario, con meno di 600 passi, sebbene l'immagine rimanga riconoscibile e poco corrotta, il numero limitato di passi nella generazione non consente alla rete di catturare completamente il rumore aggiunto nel processo di corruzione, portando a una ricostruzione incompleta.

4.3 Applicazioni: Interpolazione

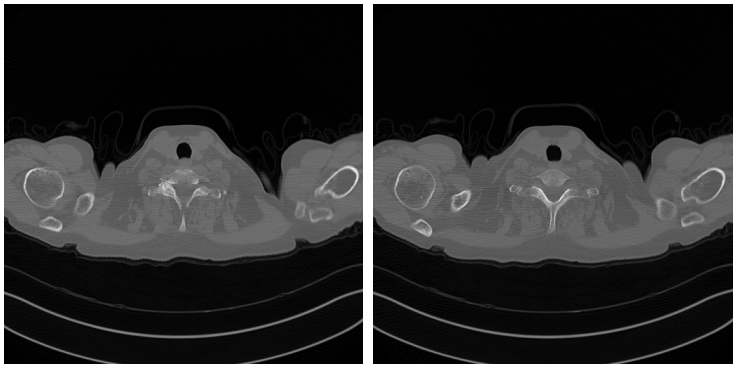
Un'applicazione interessante dell'inversione dei modelli diffusivi è l'interpolazione descritto da Xia et al. [8].

In particolare, è possibile generare un'interpolazione $\mathbf{y}_\alpha$ tra due immagini $\mathbf{y}_1$ e $\mathbf{y}_2$, modulando il parametro $\alpha \in [0, 1]$ tramite la seguente formula:

$$\mathbf{y}_\alpha = (1 - \alpha)\mathbf{y}_1 + \alpha\mathbf{y}_2 \tag{4.3}$$

Il risultato rappresenta un'immagine che combina le caratteristiche delle due immagini originali. Quando α si avvicina a 0, l'interpolazione tende a rappresentare maggiormente $\mathbf{y}_1$, mentre al tendere di α verso 1, l'immagine interpolata rappresenterà maggiormente $\mathbf{y}_2$.

Un esempio interessante è l'applicazione di questo metodo sul dataset Mayo, che contiene radiografie raffiguranti sezioni orizzontali del torace umano. Poiché il radiografo non cattura tutte le sezioni del torace, tra un'immagine e l'altra si creano delle sezioni mancanti. Utilizzando l'interpolazione, facendo variare il parametro α, è possibile generare sezioni intermedie non presenti nel dataset originale.



(a) y_1 (b) y_2

Figura 4.7: Immagini di test usate per l'interpolazione

Invece di interpolare direttamente le immagini del dataset, è possibile determinare i rumori che le generano attraverso l'inversione, interpolarli per ogni valore di α desiderato e, infine, generare una nuova immagine dal rumore interpolato. Da questo procedimento viene definto l'Algoritmo 6.

Algoritmo 6 Task di Interpolazione

function interpolation($\mathbf{y}_1$, $\mathbf{y}_2$, $\boldsymbol{\alpha}$):
  $\boldsymbol{\epsilon}_1 = \text{inversion}(\mathbf{y}_1)$
  $\boldsymbol{\epsilon}_2 = \text{inversion}(\mathbf{y}_2)$
  for α **in** $\boldsymbol{\alpha}$:
    $\boldsymbol{\epsilon}_\alpha = (1-\alpha)\boldsymbol{\epsilon}_1 + \alpha\boldsymbol{\epsilon}_2$
    $\mathbf{y}_\alpha = \text{generator}(\boldsymbol{\epsilon}_\alpha)$
  end for
  return all $\mathbf{y}_\alpha$

La funzione *generator* può essere implementata utilizzando la funzione di *sampling* dell'Algoritmo 3. Per l'inversione, si può scegliere una delle due tecniche descritte in precedenza, con diverse implicazioni in termini di risultati e prestazioni.

4.3.1 Interpolazione con Ottimizzazione

Utilizzando l'approccio basato su ottimizzazione, l'inversione può essere eseguita impostando $T = 10$ e $max\_iter = 500$. Questo approccio comporta tempi di inversione più lunghi, ma la generazione delle immagini interpolate tramite la funzione *generator* risulta più veloce, grazie al basso numero di passi di diffusione.

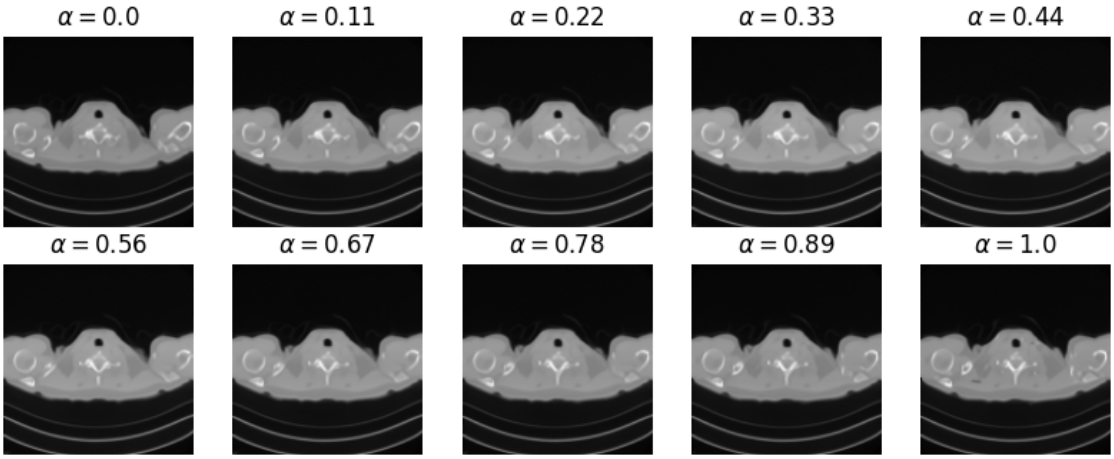


Figura 4.8: Interpolazione con ottimizzazione al variare di α

4.3.2 Interpolazione con Scheduler Inverso

Per l'interpolazione con uno scheduler inverso, possiamo utilizzare 600 passi di diffusione ($T = 600$), come discusso in precedenza, dato che questo valore ha prodotto i migliori risultati. In questo caso, l'inversione è più rapida, ma la generazione delle immagini interpolate richiede più tempo a causa del maggior numero di passi di diffusione.

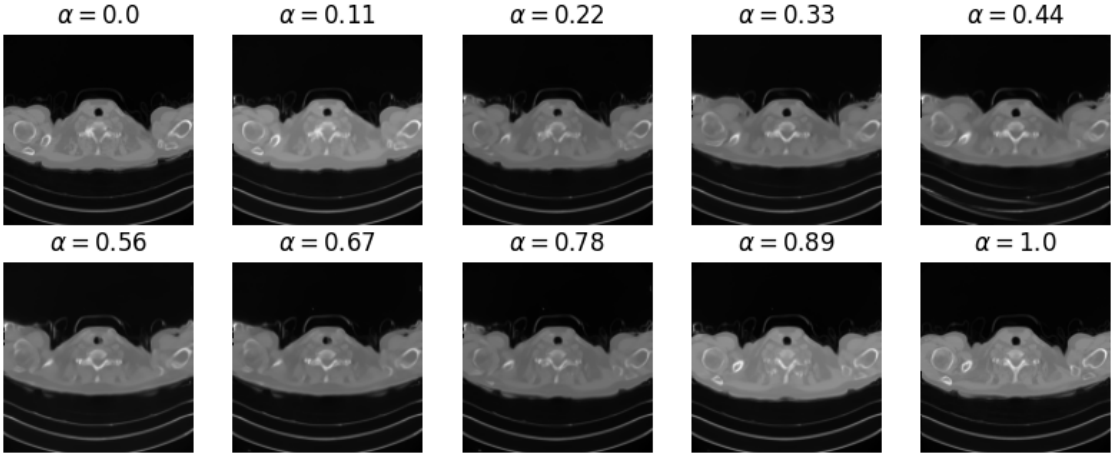


Figura 4.9: Interpolazione con scheduler inverso al variare di α

4.3.3 Confronto Metodi di Interpolazione

Dallo studio emerge un chiaro trade-off tra i due metodi di inversione. Se si desidera generare molte interpolazioni, l'approccio basato su ottimizzazione risulta più efficiente. D'altra parte, se si vuole generare poche interpolazioni tra molte immagini, l'approccio con lo scheduler inverso è più conveniente.

Di seguito è mostrato un confronto tra le immagini interpolate con α fissato a 0.44, utilizzando entrambi i metodi di inversione:

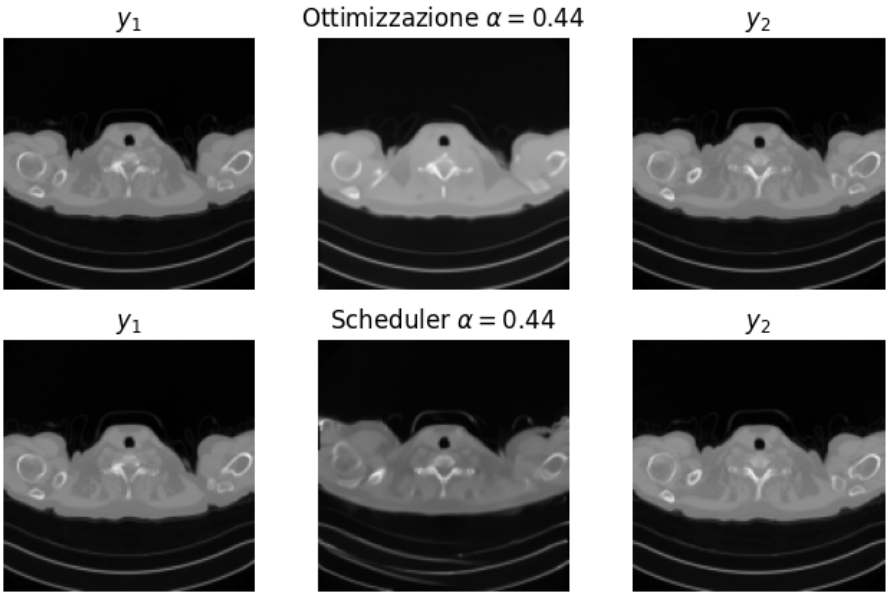


Figura 4.10: Confronto tra i metodi di interpolazione

I risultati ottenuti con i due metodi sono simili, tuttavia l'approccio con lo scheduler inverso si è dimostrato più efficace nel ricreare la parte interna della radiografia, mentre l'approccio basato sull'ottimizzazione ha mostrato superiorità nella generazione dei bordi.

Capitolo 5

Conclusioni

In questa tesi è stata approfondita l'applicazione dei modelli diffusivi nella generazione e manipolazione di immagini mediche.

I risultati ottenuti evidenziano l'efficacia di questi modelli nell'affrontare sfide significative nel campo della diagnostica medica. L'analisi dei processi di denoising e inversione ha dimostrato che i modelli diffusivi non solo migliorano la qualità delle immagini, ma offrono anche strumenti utili per il rifornimento di dati sintetici in contesti dove le tecnologie moderne non riescono a integrarsi, il tutto in tempi ragionevoli e con costi notevolmente ridotti.

Questa ricerca ha messo in luce il potenziale dei modelli diffusivi nelle immagini mediche, utilizzando tecniche fino ad ora applicate principalmente alle Generative Adversarial Network (GAN) [10], in un ambiente controllato e sostenibile.

5.1 Sviluppi Futuri

Lo studio può essere ampliato in diverse direzioni per ottimizzare ulteriormente le prestazioni dei modelli diffusivi e sfruttarne appieno il potenziale nell'elaborazione delle immagini mediche.

In primo luogo, si potrebbe esplorare l'adozione di modelli diffusivi più complessi, espandendo il modello di denoising, sperimentando con architetture avanzate e aumentando o modificando significativamente il dataset di allenamento.

Inoltre, è possibile integrare tecniche di apprendimento profondo nelle procedure di inversione o esplorare altre applicazioni che richiedono tali metodologie.

5.1.1 Inversione basata su Apprendimento

Una strategia promettente è l'addestramento di reti neurali generative, come GAN o Variational Autoencoder (VAE) [11], per apprendere direttamente il processo di inversione dai dati come descritto nell'articolo di Asperti et at. [12]. Questo approccio potrebbe portare a risultati più accurati e robusti, riducendo la dipendenza dai parametri del modello DNN.

5.1.2 Manipolazione del Rumore

Un'altra direzione interessante è l'indagine sulla manipolazione del rumore [8]. Utilizzando le tecniche di inversione, è possibile generare rumore e analizzare come la sua modifica influisca sull'immagine finale. Questo approccio può essere formalizzato matematicamente come segue:

$$\hat{\mathbf{x}} = D(\boldsymbol{\epsilon} + \alpha \mathbf{n}) \tag{5.1}$$

Qui, $\boldsymbol{\epsilon}$ rappresenta il rumore che genera un'immagine, α è un parametro che controlla l'intensità della direzione $\mathbf{n}$, e $\hat{\mathbf{x}}$ è l'immagine risultante dalla modifica del rumore.

In ambito medico, tale tecnica potrebbe essere utilizzata per simulare lo spostamento di un'immagine da una sezione del corpo all'altra.

Appendice A

Appendice

A.1 Processi Stocastici e Catene di Markov

A.1.1 Processi Stocastici

Un processo stocastico è una famiglia di variabili aleatorie indicizzate da un parametro temporale. Formalmente, un processo stocastico è una collezione di variabili aleatorie $(X_t)_{t\in T}$, dove T è l'insieme degli istanti temporali, che può essere discreto (per esempio, $T = \mathbb{N}^+$) o continuo ($T = \mathbb{R}^+$).

Questi processi possono essere descritti tramite funzioni di densità di probabilità p_X, che descrivono la probabilità di transizione tra diversi stati del processo in funzione del tempo. È inoltre possibile condizionare una variabile aleatoria rispetto agli stati precedenti, esprimendo la dipendenza dal passato del processo tramite la distribuzione condizionata:

$$p_X(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = \mathbb{P}(X_t = \mathbf{x}_t|X_{t-1} = \mathbf{x}_{t-1}, \ldots, X_0 = \mathbf{x}_0), \tag{A.1}$$

dove $(\mathbf{x}_t, \ldots, \mathbf{x}_0) \in \mathcal{X}$ e $\mathcal{X}$ rappresenta lo spazio degli stati del processo stocastico.

A.1.2 Catene di Markov

Un processo stocastico $(X_t)_{t\in T}$ è detto catena di Markov se, per ogni $t \in T$ e per ogni $(\mathbf{x}_t, \ldots, \mathbf{x}_0) \in \mathcal{X}$, vale la seguente relazione:

$$p_X(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) = p_X(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{A.2}$$

ovvero, la probabilità condizionata di X_t rispetto a X_{t-1} non dipende dagli stati precedenti. Questo implica che, data la conoscenza dello stato attuale X_{t-1}, gli stati passati

non influenzano più l'evoluzione del processo: si ha una sorta di "dimenticanza" degli stati precedenti.

Le catene di Markov presentano diverse proprietà che ne facilitano l'analisi, una delle quali riguarda la probabilità congiunta. Grazie alla proprietà di Markov, possiamo esprimere la probabilità congiunta di una sequenza di stati in modo semplificato.

Osservazione A.1.1: Probabilità Congiunta di una Catena di Markov

$$\begin{aligned} p_X(\mathbf{x}_t, \ldots, \mathbf{x}_0) &= p_X(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) p_X(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) \\ &\overset{A.2}{=} p_X(\mathbf{x}_t|\mathbf{x}_{t-1}) p_X(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_0) \\ &\ldots \\ &= \prod_{i=1}^{t} p_X(\mathbf{x}_i|\mathbf{x}_{i-1}) \cdot p_X(\mathbf{x}_0) \end{aligned}$$

Pertanto, la probabilità congiunta di una sequenza di stati in una catena di Markov può essere scritta come il prodotto delle probabilità condizionate tra stati consecutivi e la probabilità iniziale dello stato $\mathbf{x}_0$. Questo rende l'analisi delle catene di Markov particolarmente conveniente, soprattutto nei contesti in cui si devono studiare lunghe sequenze di stati.

A.2 Metriche

Le seguenti metriche sono state utilizzate per valutare la qualità dei risultati ottenuti in questo documento.

A.2.1 RMSE

La **Root Mean Squared Error** (RMSE) quantifica la differenza media quadratica tra i pixel delle immagini originale ($\mathbf{y}$) e ricostruita ($\mathbf{x}$). Più l'RMSE è bassa, maggiore è la somiglianza tra le due immagini.

$$\text{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2} \tag{A.3}$$

A.2.2 PSNR

Il **Peak Signal-to-Noise Ratio** (PSNR) valuta il rapporto tra il segnale massimo di un'immagine e il rumore che lo disturba. Questo valore, misurato in decibel, confronta le immagini $\mathbf{y}$ e $\mathbf{x}$. Un valore PSNR elevato indica una qualità di ricostruzione superiore.

$$\mathrm{PSNR}(\mathbf{x}, \mathbf{y}) = \begin{cases} 100 & \text{se } \mathrm{RMSE}(\mathbf{x}, \mathbf{y}) = 0 \\ -20 \log_{10}(\mathrm{RMSE}(\mathbf{x}, \mathbf{y})) & \text{altrimenti} \end{cases} \tag{A.4}$$

A.2.3 SSIM

Lo **Structural Similarity Index Measure** (SSIM) [13] misura la qualità visiva delle immagini in termini di percezione umana, prendendo in considerazione luminosità, contrasto e struttura. I valori del SSIM variano da -1 a 1, dove 1 indica una corrispondenza perfetta tra le immagini.

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_{\mathbf{x}}\mu_{\mathbf{y}} + C_1)(2\sigma_{\mathbf{xy}} + C_2)}{(\mu_{\mathbf{x}}^2 + \mu_{\mathbf{y}}^2 + C_1)(\sigma_{\mathbf{x}}^2 + \sigma_{\mathbf{y}}^2 + C_2)} \tag{A.5}$$

In questa formula, $\mu_{\mathbf{x}}$ e $\mu_{\mathbf{y}}$ rappresentano le medie dei pixel, $\sigma_{\mathbf{x}}$ e $\sigma_{\mathbf{y}}$ le rispettive deviazioni standard, e $\sigma_{\mathbf{xy}}$ la covarianza tra le due immagini. Le costanti C_1 e C_2 sono utilizzate per evitare divisioni per zero.

Ringraziamenti

Al termine di questo lavoro, sento il bisogno di esprimere la mia gratitudine a tutte le persone che, con il loro supporto e incoraggiamento, mi hanno permesso di raggiungere questo importante traguardo.

Innanzitutto, il mio più sentito ringraziamento va al professor Davide Evangelista, il mio relatore, per la guida preziosa, la costante disponibilità e il supporto instancabile. La sua pazienza e attenzione mi hanno accompagnato lungo tutto il percorso della stesura di questo elaborato.

Un ringraziamento alla professoressa Elena Loli Piccolomini, che mi ha aperto le porte dell'analisi numerica, accendendo in me la passione per la matematica e l'intelligenza artificiale.

Un ringraziamento speciale va alla mia famiglia, che mi ha sostenuto e incoraggiato in ogni mia decisione, rendendo possibile tutto questo.

Un ringraziamento a Becky, Sofia e Valentina, per la loro presenza costante e il supporto incondizionato che non mi hanno mai fatto mancare, facendomi sentire compreso e mai solo.

Un ringraziamnto a Gabriele, Simona e Alice, per essere stati al mio fianco lungo questo percorso, condividendo i momenti più difficili e offrendomi sempre ascolto e consigli preziosi.

Un ringraziamento a Sara, Federica, Giovanni, Gianluca, Lorenzo, Carolina, Alice, Angelo e Simone che hanno reso lo studio e le lezioni più leggere grazie ai loro sorrisi e alla loro allegria e leggerezza.

Un ringraziamento a Laura, Iman, Giulia e Samuele, che, nonostante le distanze, hanno continuato a farmi sentire il loro affetto e sostegno.

Un ringraziamento a Zaineb e Anastasia per la compagnia in treno e oltre, condividendo con me le gioie e le sfide quotidiane della vita da pendolari.

Elenco delle figure

2.1 Processi Forward e Backward di un DDPM (figura da Ho. et al [1]) . . . 7

3.1 Immagini del dataset MAYO, usate per l'allenamento del modello 12
3.2 Immagini generate attraverso la pipeline 13
3.3 Rumore Gaussiano $\sigma = 1$ 14
3.4 Rumore Speckle $\sigma = 2$ 15
3.5 Rumore Shot $\lambda = 0.25$ 16
3.6 Rumore con Scheduling Inverso - $T = 30$ 17

4.1 Risultati tramite Metodo di Ottimizzazione con $T = 10$ e $max\_it = 500$. 20
4.2 Applicazione della funzione di noising ogni 60 passi 20
4.3 Applicazione della funzione di denoising ogni 60 passi 21
4.4 Grafico di andamento delle metriche al variare di T 21
4.5 Risultato di Inversione con Immagine Generata 22
4.6 Risultato di Inversione con Immagine di Test 22
4.7 Immagini di test usate per l'interpolazione 24
4.8 Interpolazione con ottimizzazione al variare di α 25
4.9 Interpolazione con scheduler inverso al variare di α 25
4.10 Confronto tra i metodi di interpolazione 26

Elenco delle tabelle

3.1 Metriche di denoising con DDIM 17

4.1 Metriche per diversi parametri di regolarizzazione 19
4.2 Metriche di confronto tra diversi passi di diffusione T 23

Bibliografia

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[2] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[5] Verónica Bolón-Canedo, Laura Morán-Fernández, Brais Cancela, and Amparo Alonso-Betanzos. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing*, 599:128096, 2024.

[6] National Cancer Institute. Mayo dataset, n.d.

[7] Preena Prasad, Dr.J Anitha, and Divapriya Anil. A systematic review of noise types, denoising methods, and evaluation metrics in images. *2023 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, pages 1–9, 2023.

[8] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey, 2022.

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[12] Andrea Asperti, Davide Evangelista, Samuele Marro, and Fabio Merizzi. Image embedding for denoising generative models. *Artificial Intelligence Review*, 56(12):14511–14533, Dec 2023.

[13] Zhou 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*, 13(4):600–612, 2004.