Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Triennale in Informatica

Modelli Diffusivi per la Generazione di Immagini Mediche

Relatore:
Chiar.mo Prof.
Davide Evangelista

Presentata da:
Leonardo Po

Correlatrice:
Chiar.ma Prof.
Elena Loli Piccolomini

Sessione I
Anno Accademico 2024-2025

Sommario

In questa tesi viene esplorato l'impiego di modelli diffusivi per la generazione di immagini mediche. In particolare viene posta attenzione sul funzionamento dei modelli diffusivi e sulle due tipologie di architetture neurali utilizzate per implementarli.
Viene mostrato il processo di addestramento e campionamento di questi modelli su un dataset di immagini mediche, mostrando come questi siano in grado di generare immagini di ottima qualità.
Si realizza un confronto tra i modelli sviluppati, in termini di tempo di generazione e qualità della generazione.
Vengono inoltre illustrati ed analizzati i risultati dell'utilizzo di una versione sperimentale, appositamente modificata per imaging medico, delle principali metriche di valutazione per modelli generativi di immagini.

Indice

Introduzione

Negli ultimi anni, nel campo dell'intelligenza artificiale generativa, si sono affermati i **modelli a diffusione**[1] [2], modelli a variabili latenti che consentono di generare immagini di altissima qualità. Questi modelli sono in grado di generare immagini realistiche tramite un procedimento in cui si aggiunge del rumore alle immagini e si allena la rete neurale a effettuare un'operazione di rimozione del rumore. La generazione avviene facendo uso della rete neurale per rimuovere progressivamente il rumore da un campione rumoroso iniziale, fino ad ottenere una nuova immagine.
I modelli diffusivi non trovano applicazione esclusivamente nella generazione, ma vengono utilizzati in vari campi come strumenti di **ricostruzione** di immagini. In particolare, il loro utilizzo in campo medico si sta rivelando utile per migliorare la qualità delle immagini mediche che spesso sono caratterizzate dalla presenza di rumore. Ciò permette miglioramenti nell'analisi automatica delle immagini e una formulazione diagnostica più precisa da parte del personale medico.

Una problematica riguardante l'allenamento e l'utilizzo di modelli diffusivi nel campo medico è la difficoltà nell'applicazione di metriche per verificare la qualità della generazione. Le metriche più utilizzate per questo scopo sono la **Kernel Inception Distance (KID)** [3] e la **Fréchet Inception Distance (FID)** [4]: queste metriche confrontano elementi estratti da una distribuzione prodotta da una rete **InceptionV3** [5], utilizzata come estrattore di feature. La difficoltà di applicazione di queste metriche nella generazione in campo medico deriva dal fatto che il modello solitamente utilizzato come estrattore da FID e KID è allenato sul dataset ImageNet [6]. Segue che le feature estratte da InceptionV3 non rispecchiano effettivamente quelle delle immagini mediche, producendo risultati errati e non attendibili.

In questa tesi saranno prima introdotti i principi alla base del funzionamento dei modelli diffusivi. Saranno definite le architetture delle reti neurali utilizzate per i modelli diffusivi ed in seguito verrà mostrato il processo di training e campionamento di due tipologie principali di modelli diffusivi: quelli basati su **U-Net** e quelli basati su **Vision Transformer**. Verrà posta attenzione sulle generazioni effettuate dai modelli proposti e

sarà realizzato un confronto fra i modelli in termini di **qualità** delle immagini prodotte e in termini di **velocità** nella generazione.

Successivamente verrà proposta una possibile soluzione alla problematica dell'applicazione di FID e KID per imaging medico, sfruttando una versione delle metriche FID e KID in cui il modello utilizzato è una rete InceptionV3 allenata sul dataset di immagini mediche **RadImageNet** [7]. Sarà mostrato come un estrattore di feature allenato su un dataset più simile a quello utilizzato per addestrare il modello generativo che si vuole valutare, migliori di molto i valori delle metriche.

Capitolo 1

Modelli Diffusivi e Metriche di Valutazione

In questo capitolo verranno spiegati nel dettaglio i modelli diffusivi. Verrà data particolare attenzione al loro funzionamento e alle metriche che vengono utilizzate per verificare la qualità di generazione delle immagini.

1.1 Modelli generativi

Per modelli generativi si intendono modelli di intelligenza artificiale, allenati per produrre contenuti come immagini, testo, audio, video e altri media. Per farlo, si utilizza una rete neurale addestrata ad approssimare la distribuzione e i pattern presenti nei dati, che poi viene impiegata per generare contenuti completamente nuovi.

In questa tesi l'interesse sarà focalizzato sull'impiego di modelli generativi per la generazione di immagini. Esistono due tipi di generazione di immagini:

- **generazione incondizionata**: in cui i campioni vengono generati direttamente dalla distribuzione appresa dal modello, senza alcun contesto aggiuntivo dato in input ad esso.
- **generazione condizionata**: in questa tipologia di generazione l'immagine viene creata da una distribuzione condizionata da un altro input che viene passato al modello. Questo input potrebbe essere un testo o un'altra immagine. Un esempio di questa tipologia di generazione è rappresentato dai modelli Text-to-Image [8][9], che generano un'immagine a partire da un prompt testuale dato in input al modello.

I modelli impiegati per realizzare la generazione incondizionata di immagini sono i seguenti:

- **GAN**: le Generative Adversarial Networks [10] sono modelli generativi in cui vengono allenate simultaneamente due reti neurali: un **modello generativo** G che apprende la distribuzione dei dati di training, e un **discriminatore** D che viene allenato a riconoscere le immagini generate da G. G viene quindi allenato a condurre D a commettere valutazioni errate. La generazione avviene a partire da un vettore di rumore $z \sim \mathcal{N}(0, \mathbf{I})$. Il generatore G riceve in input questo vettore e restituisce in output l'immagine $G(z)$. Il modello è in grado di generare un'immagine realizzando una mappatura da $\mathcal{N}(0, \mathbf{I})$ allo spazio delle immagini. La distribuzione $\mathcal{N}(0, \mathbf{I})$ è lo spazio latente di G.

- **VAE**: i Variational AutoEncoders [11] sono modelli generativi formati da due componenti: **un encoder** E e **un decoder** D. L'encoder trasforma l'input x in un vettore latente $z \sim q(z \mid x) = \mathcal{N}(\mu(x), \sigma(x)^2)$ a dimensionalità inferiore rispetto a x. Il decoder prova a ricostruire x da z, generando $\hat{x}$. La differenza principale rispetto alle reti è GAN è che lo spazio latente $\mathcal{N}(\mu_\theta, \sigma_\theta^2)$ viene appreso dal modello tramite l'utilizzo dell'encoder. La generazione avviene fornendo un vettore $z \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$ in input a D che produce in output l'immagine $D(z)$.

- **Modelli diffusivi**: i modelli diffusivi [1] propongono un approccio diverso alla generazione, basato sulla ripetizione delle operazioni di aggiunta e rimozione di rumore gaussiano. Il modello viene allenato quindi a calcolare la quantità di rumore aggiunto a un'immagine a un determinato timestep.

  La generazione avviene ripetendo iterativamente un procedura in cui da un campione completamente rumoroso iniziale, si ottiene un'approssimazione dell'immagine finale, alla quale viene aggiunta una quantità leggermente minore di rumore gaussiano, su questo campione ottenuto viene nuovamente effettuata l'operazione sopra descritta.

1.2 Modelli diffusivi

Il funzionamento dei modelli diffusivi si fonda su due processi:

- **processo di diffusione**: consiste in una degradazione progressiva dell'immagine iniziale $\mathbf{x}_0$ tramite l'aggiunta iterativa di rumore gaussiano. Viene quindi realizzta una serie di spazi latenti aventi la medesima dimensionalità di $\mathbf{x}_0$: $\mathbf{x}_1, \ldots, \mathbf{x}_T$, dove T è il numero di iterazioni.

- **processo inverso**: il processo inverso ha lo scopo di invertire l'aggiunta di rumore in più passi realizzata dal processo di diffusione. Avviene tramite l'ausilio di una rete neurale, ed è il responsabile della generazione delle immagini. Effettua

un'operazione di denoising in più passi, il cui numero di passi può essere uguale a quello utilizzato per il processo di diffusione (DDPM) o minore (DDIM).

In questa sezione verranno approfonditi i dettagli implementativi e matematici alla base dei modelli diffusivi e del loro funzionamento. Esistono due tipologie principali di modelli diffusivi:

- **DDPM**: i *Deep Denoising Probabilistic Models* [1] sono i primi modelli diffusivi, si caratterizzano per un processo inverso stocastico.
- **DDIM**: i *Deep Denoising Implicit Models* [2] sono caratterizzati da un processo inverso deterministico, che consente una generazione di immagini molto più veloce rispetto ai DDPM.

1.2.1 DDPM

Si consideri la distribuzione $q(\mathbf{x}_0)$ dei dati di training, si vuole realizzare un'approssimazione $p(\mathbf{x}_0)$. Ciò avviene integrando le variabili latenti $\mathbf{x}_1, ..., \mathbf{x}_T$, descritte all'inizio di questa sezione:

$$p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{1:T}, \quad \text{dove} \quad p_\theta(\mathbf{x}_{0:T}) := p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \tag{1.1}$$

dove θ rappresenta i parametri allenabili del modello.

In figura 1.1 viene mostrato il funzionamento di un modello DDPM.

Forward Process

Il forward process (processo di diffusione) consiste nella realizzazione dello spazio latente da cui il modello dovrà ricostruire le immagini. Come anticipato precedentemente in questa sezione, realizza una graduale aggiunta di rumore gaussiano all'immagine iniziale, fino a ottenere una completa degradazione $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Il forward process è definito da una **catena di Markov** con parametri $\alpha_{1...T} \in (0, 1]$ tali che $\forall i(\alpha_{i+1} \leq \alpha_i)$:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \quad \text{dove} \quad q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\left(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\mathbf{x}_{t-1}, \left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right)\mathbf{I}\right) \tag{1.2}$$

Qui si osservano due differenze rispetto agli altri modelli generativi: lo spazio latente è ottenuto tramite una procedura fissata (non allenabile) e ha la stessa dimensionalità delle immagini di input.

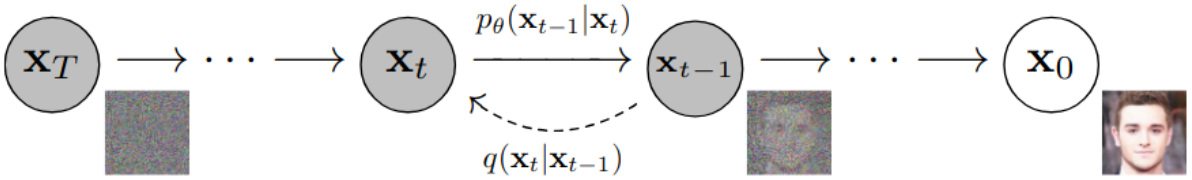

Figura 1.1: Illustrazione di forward e backward dal paper di DDPM [1]

Una proprietà fondamentale del processo di noising è che la distribuzione marginale $q(\mathbf{x}_t \mid \mathbf{x}_0)$ è calcolabile come segue:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) := \int q(\mathbf{x}_{1:t} \mid \mathbf{x}_0)\, d\mathbf{x}_{1:t-1} = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I}) \tag{1.3}$$

segue che $\mathbf{x}_t$ è definibile tramite una combinazione lineare fra $\mathbf{x}_0$ e una variabile ϵ di rumore gaussiano .

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon, \quad \text{dove } \epsilon \sim \mathcal{N}(0, I) \tag{1.4}$$

Backward Process

Il processo inverso è quello che viene realizzato da una rete neurale che effettua denoising. L'obiettivo è quello di ottenere una ricostruzione di $\mathbf{x}_0$ a partire da $\mathbf{x}_T$. Il processo inverso è dato dalla distribuzione $p_\theta(\mathbf{x}_{0:T})$. In particolare si vuole andare a invertire il forward process, per trovare un'approssimazione di $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Nei DDPM il backward process è un processo stocastico, modellato attraverso una catena di Markov:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \tag{1.5}$$

ciò consente di dare la definizione di $p_\theta(\mathbf{x}_{0:T})$ sottostante:

$$p_\theta(\mathbf{x}_{0:T}) := p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta^{(t)}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \tag{1.6}$$

Durante il training il modello scelto, che potrà essere una U-Net o un Transformer, modificherà i propri parametri θ al fine di calcolare le approssimazioni della media $\mu_\theta(\mathbf{x}_t, t)$ e della varianza $\Sigma_\theta(\mathbf{x}_t, t)$ della quantità di rumore applicata al timestep t all'immagine.

Training

Calcolare direttamente l'inverso del forward process facendo uso della probabilità a posteriori $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ è impossibile, in quanto applicando il Teorema di Bayes sarebbe richiesto il calcolo di $q(\mathbf{x}_t) = \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{0:t-1}\, d\mathbf{x}_{t+1:T}$ e di $q(\mathbf{x}_{t-1}) = \int p_\theta(\mathbf{x}_{0:T})\, d\mathbf{x}_{0:t-2}\, d\mathbf{x}_{t:T}$ che sono intrattabili poiché richiederebbero l'utilizzo dell'intero dataset. Si procede quindi calcolando $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$ attraverso l'ausilio di una DDN (*Deep Denoising Network*). La media $\mu_\theta(\mathbf{x}_t, t)$ e la deviazione standard $\Sigma_\theta(\mathbf{x}_t, t)$ hanno la seguente formulazione:

$$\mu_\theta(\mathbf{x}_t, t) = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \frac{\alpha_t}{\alpha_{t-1}}}{\sqrt{1-\alpha_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \tag{1.7}$$

$$\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I} = \left(1 - \frac{\alpha_t}{\alpha_{t-1}} \right) \mathbf{I} \tag{1.8}$$

Si nota subito che la varianza $\Sigma_\theta(\mathbf{x}_t, t)$ non dipende dai parametri della rete θ. Questo permette di definire la funzione di loss $L(\theta)$ sottostante:

$$L(\theta) := \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\alpha_{t-1} - \alpha_t}{2\alpha_t(1-\alpha_t)} \left|\left| \epsilon - \epsilon_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon, t) \right|\right|^2 \right] \tag{1.9}$$

Questa funzione di loss è già utilizzabile per il training, tuttavia è possibile realizzare ulteriori semplificazioni, che portano a definire:

$$L(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\left|\left| \epsilon - \epsilon_\theta\big(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon, t\big) \right|\right|^2 \right] \qquad t \sim U(1, ..., T) \tag{1.10}$$

Algorithm 1 Training di DDPM e DDIM

while not `convergenza` **do**
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim U(\{1, \ldots, T\})$
  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$
  Calcola backpropagation su
  $\nabla_\theta \left|\left| \epsilon - \epsilon_\theta(\mathbf{x}_t, t) \right|\right|$
end while

Sampling

Il processo di sampling (mostrato in figura 1.2) sfrutta il modello allenato a effettuare denoising, per calcolare un campione $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ a partire da un campione $\mathbf{x}_t$.

$$\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \frac{\alpha_t}{\alpha_{t-1}}}{\sqrt{1-\alpha_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}. \tag{1.11}$$

In questa equazione $\frac{\epsilon_\theta(\mathbf{x}_t,t)}{\sqrt{1-\alpha_t}}$ è un'approssimazione stimata dalla rete della quantità di rumore che è stata aggiunta a $\mathbf{x}_0$ per ottenere $\mathbf{x}_t$ ($\epsilon_\theta(\mathbf{x}_t, t)$ è un'approssimazione del rumore ϵ presente in equazione (1.4)). Eseguendo il prodotto tra questa quantità e $1-\frac{\alpha_t}{\alpha_{t-1}}$ si ottiene una stima della quantità di rumore aggiunta a $\mathbf{x}_{t-1}$ per ottenere $\mathbf{x}_t$. Infine la moltiplicazione per il fattore $\sqrt{\frac{\alpha_{t-1}}{\alpha_t}}$ consente di calcolare l'approssimazione per $\mathbf{x}_{t-1}$, alla quale può essere aggiunto un nuovo rumore random $\sigma_t \mathbf{z}$, dove $\sigma_t^2 = (1 - \frac{\alpha_t}{\alpha_{t-1}})$. Questa operazione può essere eseguita per ogni $t \geq 2$, poiché per $t = 1$, se si andasse ad applicare un ulteriore rumore gaussiano si andrebbe ad aggiungere rumore alla ricostruzione finale dell'immagine, compromettendo la qualità della generazione.

Algorithm 2 Sampling con DDPM

$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
for $t = T, \ldots, 1$ **do**:
  if $t > 1$ **then**
    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
  else
    $\mathbf{z} = 0$
  end if
  $\mathbf{x}_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\frac{\alpha_t}{\alpha_{t-1}}}{\sqrt{1-\alpha_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}.$
end for
return $\mathbf{x}_0$

1.2.2 DDIM

Nonostante i DDPM consentano la generazione di campioni di elevata qualità, i tempi di generazione sono tendenzialmente molto lunghi. Questa lentezza computazionale è dovuta al numero elevato di step di denoising necessari per la rimozione del rumore (solitamente $\mathbf{T} = 1000$). I Deep Denoising Implicit Models sono caratterizzati da un processo di generazione deterministico, quindi non modellato da una catena di Markov (processo stocastico). Questo consente la realizzazione di un processo inverso di durata fortemente ridotta rispetto ai DDPM, senza che i campioni generati perdano in qualità.

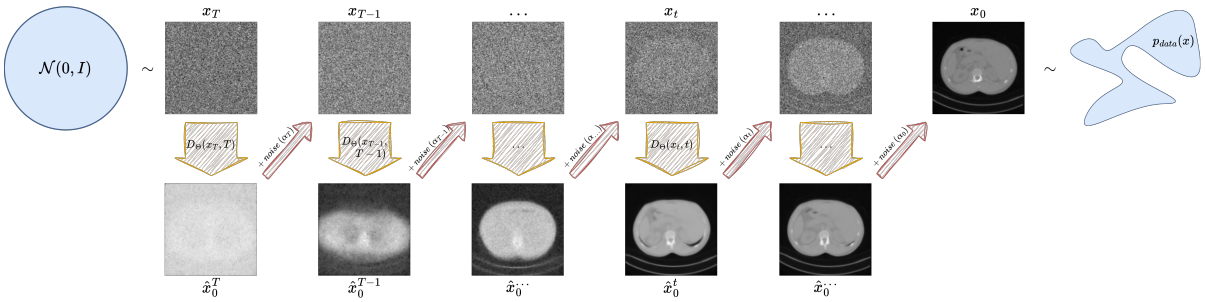

Figura 1.2: Procedura di sampling

Forward process

Il forward process nella pratica rimane uguale a DDPM, tuttavia in [2] viene formulato un processo di diffusione non markoviano, definito nel seguente modo:

$$q_\sigma(\mathbf{x}_{1:T}|\mathbf{x}_0) := q_\sigma(\mathbf{x}_T|\mathbf{x}_0) \prod_{t=2}^{T} q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \tag{1.12}$$

dove $q_\sigma(\mathbf{x}_T \mid \mathbf{x}_0) = \mathcal{N}(\sqrt{\alpha_T}\mathbf{x}_0, (1-\alpha_T)\mathbf{I})$ e per ogni $t > 1$

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2\mathbf{I}\right). \tag{1.13}$$

Nonostante questa formulazione non markoviana del processo di diffusione, è dimostrabile che la proprietà $q(\mathbf{x}_t \mid \mathbf{x}_0) := \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I})$ continua a rimanere valida.

Il forward process $q_\sigma(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0)$ è facilmente ricavabile:

$$
\begin{aligned}
&q_\sigma(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) \\
&\qquad = \frac{q_\sigma(\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_0)}{q(\mathbf{x}_{t-1}, \mathbf{x}_0)} \\
&\qquad = \frac{q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) q_\sigma(\mathbf{x}_t, \mathbf{x}_0)}{q(\mathbf{x}_{t-1}, \mathbf{x}_0)} \\
&\qquad = \frac{q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) q_\sigma(\mathbf{x}_t \mid \mathbf{x}_0) q_\sigma(\mathbf{x}_0)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) q_\sigma(\mathbf{x}_0)} \\
&\qquad = \frac{q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) q_\sigma(\mathbf{x}_t \mid \mathbf{x}_0)}{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}
\end{aligned}
$$

Il valore di σ modula la stocasticità del processo di aggiunta del rumore. Per $\sigma \to 0$ si raggiunge una casistica nella quale, conoscendo $\mathbf{x}_t$ e $\mathbf{x}_0$, il campione $\mathbf{x}_{t-1}$ diventa fisso, rendendo il processo di diffusione deterministico.

Backward Process

La possibilità di definire un forward process non markoviano consente quindi di invertire questo processo con un processo inverso anch'esso non markoviano. Il processo generativo è dato dalla distribuzione congiunta $p_\theta(\mathbf{x}_{0:T})$, dove $p_\theta^{(t)}(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ vuole ricostruire $q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Dall'osservazione rumorosa iniziale $\mathbf{x}_t$ si ottiene una previsione di $\mathbf{x}_0$, che viene utilizzata per ricostruire $\mathbf{x}_{t-1}$ attraverso la distribuzione $q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$:

$$
p_\theta^{(t)}(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \begin{cases} \mathcal{N}(f_\theta(\mathbf{x}_1, 1), \sigma_1^2 I) & \text{se } t = 1 \\ q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, f_\theta(\mathbf{x}_t, t)) & \text{altrimenti} \end{cases} \tag{1.14}
$$

dove $f_\theta(\mathbf{x}_t, t)$ è una funzione che restituisce un'approssimazione di $\mathbf{x}_0$ dato $\mathbf{x}_t$ tramite la previsione del rumore calcolata dal modello $\epsilon_\theta(\mathbf{x}_t, t)$:

$$
f_\theta(\mathbf{x}_t, t) := (\mathbf{x}_t - \sqrt{1 - \alpha_t} \cdot \epsilon_\theta(\mathbf{x}_t, t))/\sqrt{\alpha_t}. \tag{1.15}
$$

Si noti che $f_\theta(\mathbf{x}_t, t)$ è ottenuta dalla formula inversa dell'equazione (1.4).

Training

Il training di DDIM rimane identico al training di un modello DDPM. La funzione di loss, quindi, è analoga a quella definita dall'equazione (1.10).

Sampling

La generazione di un campione $\mathbf{x}_{t-1}$ da un campione $\mathbf{x}_t$ avviene attraverso:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} \right)}_{\text{approssimazione di } \mathbf{x}_0} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(\mathbf{x}_t, t)}_{\text{direzione verso } \mathbf{x}_t} + \underbrace{\sigma_t \epsilon_t}_{\text{rumore aggiuntivo}} \tag{1.16}$$

Il termine $\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(\mathbf{x}_t, t)$ regola il rumore aggiunto all'approssimazione di $\mathbf{x}_0$ per ottenere $\mathbf{x}_{t-1}$; questo avviene moltiplicando la previsione del rumore per il fattore $\sqrt{1-\alpha_t-\sigma_t^2}$ che indica la quantità di rumore da applicare al timestep $t-1$. Inoltre, è possibile rendere il processo stocastico tramite l'aggiunta di rumore casuale $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$ non dipendente dal modello.

Si nota che la stocasticità è completamente regolata dal termine σ_t, se per ogni t si definisce $\sigma_t = 0$ si ottiene un forward process completamente deterministico (DDIM).

Generazione velocizzata

Dal momento che il processo di campionamento non è più markoviano, non si rende necessario compiere un elevato numero di passi per avere una generazione di qualità. Non serve utilizzare tutte le variabili latenti $\mathbf{x}_{1:\mathbf{T}}$, ma è sufficiente definire il backward process su un sottoinsieme $\{\mathbf{x}_{\tau_1}, \dots \mathbf{x}_{\tau_s}\}$ con $\tau \subset [1, \dots, \mathbf{T}]$ di lunghezza s. La procedura di sampling rimane analoga a quella vista precedentemente:

$$\mathbf{x}_{\tau_{i-1}} = \sqrt{\alpha_{\tau_{i-1}}} \left(\frac{\mathbf{x}_{\tau_i} - \sqrt{1-\alpha_{\tau_i}}\epsilon_\theta(\mathbf{x}_{\tau_i}, \tau_i)}{\sqrt{\alpha_{\tau_i}}} \right) + \sqrt{1-\alpha_{\tau_{i-1}}} \cdot \epsilon_\theta(\mathbf{x}_{\tau_i}, \tau_i) \tag{1.17}$$

dove i è l'i-esimo elemento della sottosequenza di timestep definita da τ.

In figura 1.3 è possibile visualizzare la procedura di generazione velocizzata appena descritta.

Algorithm 3 Sampling velocizzato con DDIM

$\mathbf{x}_{\tau_s} \sim \mathcal{N}(0, \mathbf{I})$
for $i = s, \dots, 1$ **do**:
  $\mathbf{x}_{\tau_{i-1}} = \sqrt{\alpha_{\tau_{i-1}}} \left(\frac{\mathbf{x}_{\tau_i} - \sqrt{1-\alpha_{\tau_i}}\epsilon_\theta(\mathbf{x}_{\tau_i}, \tau_i)}{\sqrt{\alpha_{\tau_i}}} \right) + \sqrt{1-\alpha_{\tau_{i-1}}} \cdot \epsilon_\theta(\mathbf{x}_{\tau_i}, \tau_i)$
end for
return $\mathbf{x}_0$

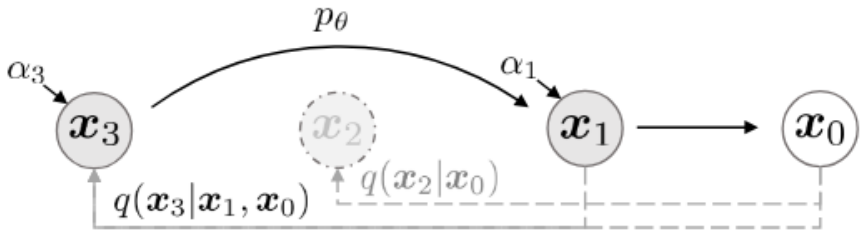

Figura 1.3: Illustrazione del procedura di generazione velocizzata dal paper di DDIM [2]

1.3 Architetture per modelli diffusivi

Per modello a diffusione non si intende uno specifico modello, con una determinata architettura sottostante; la diffusione riguarda le modalità di allenamento e campionamento impiegate. Nelle implementazioni originali di DDPM [1] e DDIM [2] l'architettura utilizzata è una U-Net [12]. Con l'arrivo delle reti neurali basate su architettura Transformer [13], delle applicazioni di quest'ultime a task di computer vision [14], è stata proposta anche l'architettura U-ViT [15], un diffusion model basato su architettura ViT.

1.3.1 U-Net

U-Net è una rete neurale profonda (mostra in figura 1.4) proposta nel 2015 [12] per effettuare la segmentazione di immagini biomediche. Può essere facilmente adattata con successo a qualsiasi task di tipo image-to-image. Il nome deriva dalla sua forma a "U", composta da due elementi principali:

- **percorso di contrazione**: è la parte sinistra della rete, costituita da blocchi convoluzionali in cui si alterna l'applicazione di convoluzioni 3×3 e layer di max pooling 2×2 per effettuare l'estrazione delle feature principali. A ogni passo di downsampling, il numero dei canali aumenta. Questa prozione della rete si comporta come un encoder.

- **percorso di espansione**: è la parte destra della rete, è formata anch'essa da blocchi convoluzionali, caratterizzati dall'alternanza di layer di convoluzione trasposta 2×2 e di layer convoluzionali 3×3. L'obbiettivo di questa porzione dell'architettura è quello di effettuare un'operazione di up-sampling dalle informazioni estratte dall'encoder. Per consentire alla rete di recuperare eventuali informazioni spaziali sulle immagini, perse durante l'encoding, viene fatto uso di skip connections in cui vengono concatenate le feature map corrispondenti che provengono dal percorso di contrazione. Il percorso di espansione agisce come un decoder.

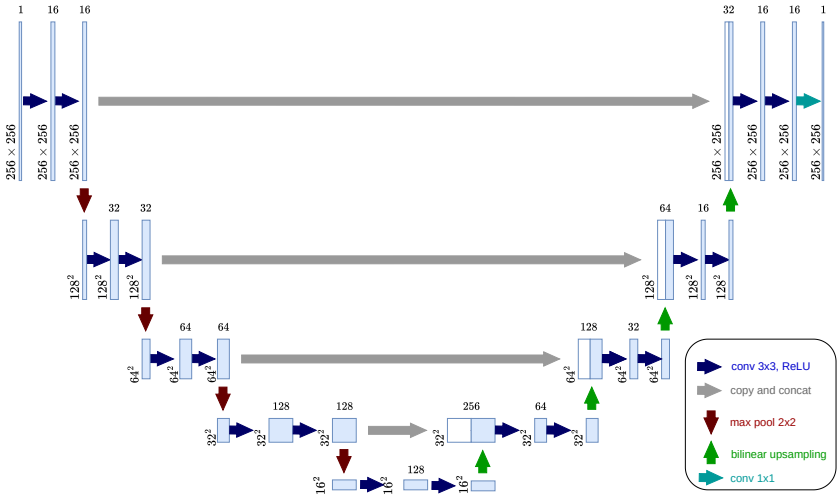

Figura 1.4: Esempio di architettura di una rete U-Net

Meccanismo di attenzione

Il meccanismo di attenzione [13] è il fulcro delle reti Transformer, in quanto consente al modello di apprendere le relazioni contestuali presenti fra i vari token.

Il meccanismo di attenzione consiste nel calcolo di una somma pesata che consente di individuare i token più importanti e la loro relazione con gli altri elementi della sequenza. In particolare dai token $T \in \mathbb{R}^{l \times d}$ vengono calcolate delle matrici di dimensione minore dette *key, query, value* $Q, K, V \in \mathbb{R}^{l \times d_k}$:

- Ogni riga della matrice Q può essere vista come il token della sequenza che si sta osservando in un istante.
- Ogni riga della matrice K identifica i token della sequenza consentendo il confronto con la query.
- Ogni riga della matrice V è un indicatore del valore (importanza) di un determinato token in caso di forte corrispondenza tra q e k dove q e k sono rispettivamente righe di Q e K.

dove d è la lunghezza di un embedding e d_k è data da d/h con h numero di teste di attenzione, ovvero quante triple di matrici Q, K, V vengono utilizzate. Il calcolo avviene tramite la seguente formula:

$$\text{Attenzione}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.18}$$

dove la funzione $\text{softmax}(\mathbf{X})_{ij} = \frac{e^{X_{ij}}}{\sum_{k=1}^{m} e^{X_{ik}}}$ trasforma i valori in un range compreso tra 0 e 1, realizzando una distribuzione di probabilità.

Dato che si stanno usando più teste di attenzione, si rende necessario unire e ridimensionare l'output delle varie teste:

$$\text{MultiAttenzione}(Q, K, V) = \text{Concat}(h_1, \ldots, h_h)W^O \quad \text{dove } h_i = \text{Attenzione}(Q_i, K_i, V_i)$$

Dove $W^O \in \mathbb{R}^{(hd_k)\times d}$ è una matrice di parametri del modello.

Layer Convolutivo con Attenzione

Considerando la classica operazione di convoluzione si ha:

- il kernel $K \in \mathbb{R}^{C_{out}\times C_{in}\times k\times k}$ con k kernel size e C_{in} e C_{out} numero di canali di input e di output
- il tensore di input $F \in \mathbb{R}^{C_{in}\times H\times W}$
- il tensore di output $G \in \mathbb{R}^{C_{out}\times H\times W}$

L'operazione di convoluzione che da F porta a ottenere G tramite K, è definita attraverso la seguente formulazione:

$$g_{ij} = \sum_{p,q} K_{p,q} f_{i+p-\lfloor k/2\rfloor, j+q-\lfloor k/2\rfloor} \tag{1.19}$$

dove $g_{i,j} \in \mathbb{R}^{C_{out}}$ e $f_{ij} \in \mathbb{R}^{C_{in}}$ sono i vettori di feature associati al bit in posizione (i, j) del tensore di output e di input. Gli indici $p, q \in \{0, \ldots, k-1\}$ sono gli indici della posizione del kernel.

Il meccanismo di self-attention [13] nelle reti convolutive consiste nell'applicare un blocco di attenzione in seguito a un blocco convolutivo classico che compie l'operazione definita in (1.19). Questo consente di catturare, oltre alle caratteristiche locali, anche le caratteristiche globali dell'immagine.

Ciò avviene calcolando l'attenzione su un tensore in cui l'asse dell'altezza e l'asse della larghezza sono stati appiattiti in un unico asse (da $G \in \mathbb{R}^{C_{out}\times H\times W}$ a $G' \in \mathbb{R}^{C_{out}\times HW}$). Il calcolo del meccanismo di attenzione viene mostrato in sezione 1.3.1.

Timestep Embedding

Nei modelli diffusivi la rete viene allenata a effettuare una previsione del rumore introdotto fino al tempo t; di conseguenza diventa necessario rendere il modello consapevole del tempo t al quale sta operando. Questo avviene grazie agli embedding, che permettono al modello di adattare la propria azione in base al valore di t, in modo da predire correttamente $\epsilon_\sigma(\mathbf{x}_t, t)$.

Il calcolo dei timestep embedding avviene come segue:

$$f_j = \exp\left(\frac{(\ln 10000) \cdot j}{dim/2}\right) \quad \forall j \in \{0, \ldots, dim/2 - 1\}$$

$$A_{i,j} = t_{i,1} \cdot f_{1,j} \quad \forall j \in \{0, \ldots, dim/2 - 1\}, \forall i \in \{0, \ldots, n-1\}$$

$$E = [\cos(A), sin(A)]$$

dove:

- dim è la lunghezza di un vettore di embedding;
- n è la lunghezza del batch preso in considerazione;
- $f \in \mathbb{R}^{1\times(dim/2)}$ è il vettore delle frequenze;
- $t \in \mathbb{R}^{n\times 1}$ è il vettore contenente i valori dei timestep associati ai campioni del batch;
- $A \in \mathbb{R}^{n\times(dim/2)}$ è la matrice dei prodotti fra timestep e frequenze;
- $E \in \mathbb{R}^{n\times dim}$ è la matrice finale degli embedding.

Le informazioni dell'embedding vengono poi mappate a una dimensione pari al numero di canali di output di un blocco convolutivo tramite MLP (*Multilayer Perceptron*), consentendo la somma con l'output del blocco convolutivo.

1.3.2 U-ViT

L'architettura U-ViT introdotta nel 2023 [15] offre un'alternativa agli approcci basati su architetture convoluzionali. Si ispira all'architettura ViT [14], la quale fa uso di una rete basata su Transformer per effettuare la classificazione di immagini.

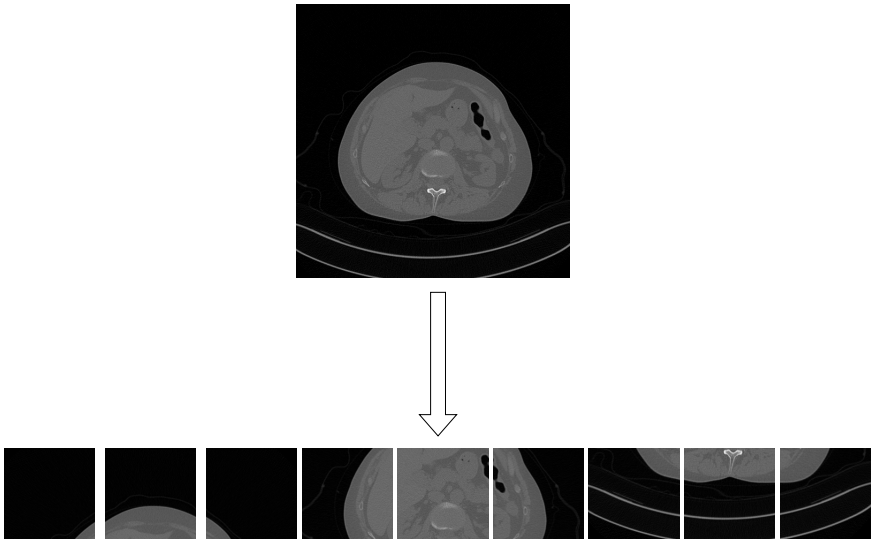

Figura 1.5: Illustrazione semplificata della procedura di *patching*

Patch Embedding

Le reti basate su Transformer [13] offrono performance ben superiori rispetto ad altre architetture per effettuare encoding e generazione di dati sequenziali. Nonostante gli eccezionali risultati in task di elaborazione del linguaggio naturale, la morfologia di una rete Transformer non permette al modello di lavorare direttamente con le immagini. Si rende quindi necessaria un'operazione detta *patching*, finalizzata a trasformare un'immagine in una **sequenza** di *patch* caratterizzati da una risoluzione inferiore (mostrata in figura 1.5).

Ad esempio, un'immagine 256×256 viene trasformata in una sequenza di 256 patch di risoluzione 16×16.

In seguito i patch vengono appiattiti e trasformati nella dimensione decisa per i token. Ogni patch viene quindi trasformato in un token.

Timestep Embedding

Il timestep embedding è definito come nella sezione 1.3.1 e il token associato viene aggiunto all'inizio della sequenza dei token ricavati dai patch.

Architettura

L'architettura U-ViT fa uso di un numero pari di blocchi Transformer, applicati dopo che dai token di input è stato calcolato il positional embedding. Per **positional embedding**

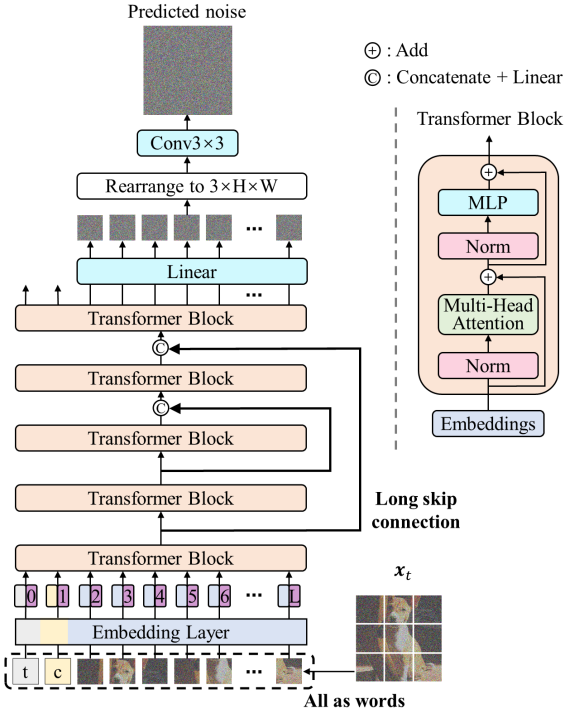

Figura 1.6: Immagine dell'architettura di U-ViT dal rispettivo articolo [15]

si intende un insieme di parametri della rete, che vengono aggiornati al fine di conferire al token informazioni sulla posizione del patch all'interno dell'immagine in input.

I blocchi utilizzati da U-ViT sono blocchi di **encoding**, disposti simmetricamente in una modalità che ricorda U-Net (da qui la "U" in U-ViT). Come avviene in U-Net, la seconda metà dei blocchi riceve in input anche una skip connection, consistente nei token di output provenienti dal blocco in posizione simmetrica rispetto a quello corrente. I token di input e quelli provenienti dalla skip connection vengono concatenati lungo l'asse degli embedding e successivamente mappati alla dimensione originale degli embedding attraverso un livello completamente connesso. I **blocchi Transformer** sono costituiti da un blocco di **attenzione** (definito in 1.3.1) e da un blocco **MLP** (*Multilayer Perceptron*) consistente in due livelli totalmente connessi, in cui il prodotto del blocco di attenzione viene proiettato in uno spazio a dimensionalità maggiore Ciò permette al modello di apprendere più connessioni fra i token, migliorando la rappresentazione di essi. In seguito la dimensionalità aumentata del livello nascosto viene riportata a quella iniziale. L'output del blocco MLP diventerà l'input del blocco Transformer successivo.

In figura 1.6 è possibile osservare l'architettura originale di U-ViT proposta in [15].

1.4 Embedder

La valutazione della qualità di un modello generativo di immagini è diversa dalla maggior parte delle valutazioni di altre task di apprendimento automatico. La differenza principale risiede nell'**assenza di una ground truth** di riferimento che consenta di calcolare una metrica di confronto diretto (come avviene con metriche come MSE e $RMSE$ per task di ricostruzione di immagini).

Per questa ragione vengono utilizzate metriche che fanno uso di modelli **embedder**, che estraggono le caratteristiche delle immagini ricevute in input. Per **embedder** si intendono modelli di machine learning capaci di produrre una rappresentazione più efficiente della complessità dei dati in input, solitamente vettoriale, che può essere poi utilizzata da altri modelli per effettuare le proprie task con maggiore facilità. L'aspetto più importante degli embedding è che, nonostante consentano una rappresentazione più efficiente del dato iniziale, mantengono intatte la maggior parte delle relazioni semantiche presenti nel dato.

Esistono varie tipologie di embedder, in seguto sono riportati alcuni esempi:

- **Blocco Transformer di encoding**: Il blocco di encoding dell'architettura Transformer [13] effettua un embedding in profondità grazie alla rete neurale, consentendo di rappresentare ogni token con un embedding di lunghezza predefinita. Questa tipologia è ideale per problemi di elaborazione del linguaggio naturale.
- **Embedder non allenabili**: tecniche come **PCA** (*Principal Component Analysis*) e **SVD** (*Singular Value Decomposition*) consentono di ridurre la dimensionalità del dato iniziale, perdendo una ridotta quantità di contenuto informativo, attraverso l'utilizzo di operazioni dell'algebra lineare. Essendo queste tecniche basate su trasformazioni lineari, vengono solitamente usate su dati numerici che presentano delle relazioni lineari.
- **Embedder per immagini**: consistono in reti neurali convolutive molto profonde, solitamente allenate per task di classificazione su dataset molto ampi come ImageNet. Da queste reti si rimuove la porzione finale di livelli totalmente connessi, utilizzati per la classificazione. L'output dell'embedder sarà quindi un vettore a dimensionalità ridotta delle caratteristiche estratte dall'immagine. Le metriche FID e KID usano questa tipologia di embedder.

L'utilizzo di un embedder è necessario per poter calcolare le metriche FID e KID, infatti adattare queste metriche per essere calcolate direttamente sulle immagini significherebbe confrontare vettori di lunghezza $299 \times 299 \times 3$ invece di 2048 (nel caso di InceptionV3). Le immagini generate, se il training del modello è stato effettivo, avranno pattern e caratteristiche simili alle immagini utilizzate nel dataset. Questo significa che le distribuzioni delle feature estratte saranno simili.

Il modello maggiormente utilizzato come estrattore di feature è **InceptionV3** [5], basato su un'architettura facente uso di blocchi convoluzionali che applicano in parallelo convoluzioni con kernel di varie dimensioni per permettere un'estrazione capillare delle caratteristiche visive dell'immagine. In figura 1.7 si può osservare una raffigurazione ad alto livello dell'architettura di InceptionV3.

1.5 Metriche

In questa sezione vengono definite le metriche utilizzate per verificare la qualità del sampling di un modello generativo. Le due metriche principali utilizzate sono:

- **FID**: *Fréchet Inception Distance* [4]
- **KID**: *Kernel Inception DIstance* [3]

1.5.1 FID

La metrica FID è matematicamente definita come segue:

$$\text{FID} = ||\mu - \mu_w||^2 + tr(\Sigma + \Sigma_w - \Sigma(\Sigma\Sigma_w)^{\frac{1}{2}}) \tag{1.20}$$

Nell'equazione soprastante $\mathcal{N}(\mu, \Sigma)$ è una distribuzione normale multivariata stimata dalla rete neurale InceptionV3 [5] sulle immagini reali (non generate dal modello generativo in esame). $\mathcal{N}(\mu_w, \Sigma_w)$ è invece la distribuzione normale multivariata stimata dalle immagini prodotte da un modello generativo. $tr(\Sigma + \Sigma_w - \Sigma(\Sigma\Sigma_w)^{\frac{1}{2}})$ è la traccia della matrice $\Sigma + \Sigma_w - \Sigma(\Sigma\Sigma_w)^{\frac{1}{2}}$. La FID può essere quindi vista come una misura della distanza di due distribuzioni normali multivariate prodotte dall'ultimo layer di pooling di un modello InceptionV3.

Si parla di **distribuzioni normali multivariate** perché la distanza di Fréchet è una distanza pensata per misurare quanto due curve differiscono fra loro. Si assume che le distribuzioni prodotte da InceptionV3 siano distribuzioni normali, anche se non è detto che sia così.

1.5.2 KID

La Kernel Inception Distance (KID), come la metrica FID, è una metrica per valutare la qualità dei modelli generativi. Misura la dissomiglianza tra due distribuzioni di immagini calcolando una stima empirica della **SMMD** (*Squared Maximum Mean Discrepancy*) tra le feature estratte dalle immagini attraverso InceptionV3.

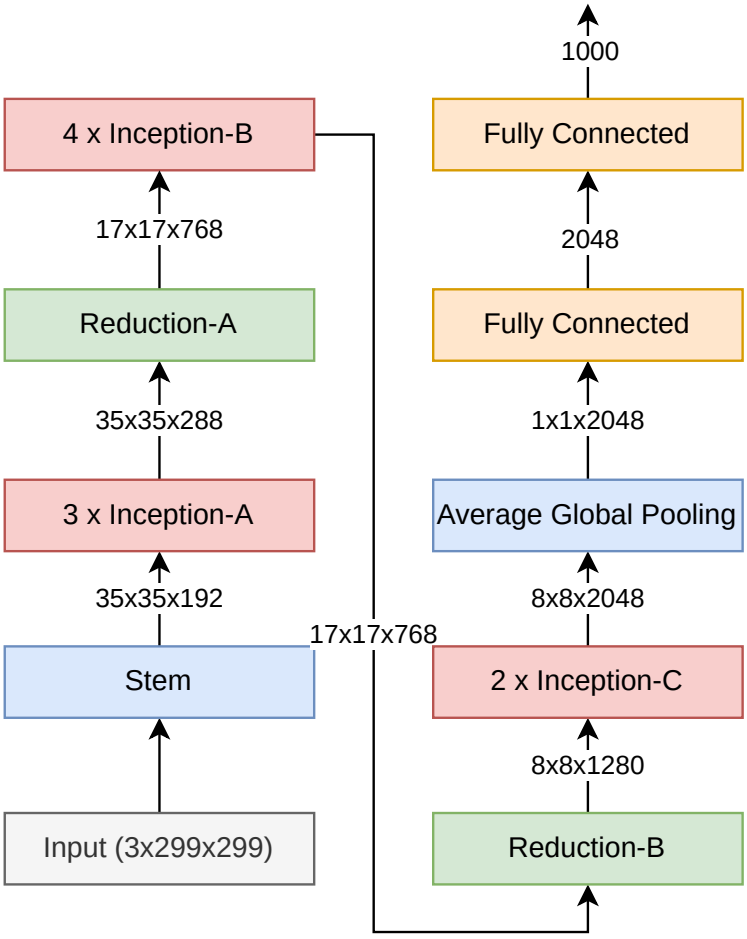

Figura 1.7: Architettura di InceptionV3

La stima empirica di SMMD viene calcolata attraverso la seguente equazione, dove $\{x_i\}_{i=1}^m \sim P$, $\{y_j\}_{j=1}^n \sim Q$, sono campioni delle distribuzioni P e Q prodotte da InceptionV3:

$$\begin{aligned} \text{SMMD} = (&\frac{1}{m(m-1)} \sum_{i \neq j} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i \neq j} k(y_i, y_j) \\ &- \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x_i, y_j))^2 \end{aligned} \tag{1.21}$$

Nell'equazione sovrastante $k : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ è un kernel polinomiale di grado 3:

$$k(x, y) = \left(\frac{1}{d} x^T y + 1\right)^3$$

dove:

- $x, y \in \mathbb{R}^d$ sono vettori di feature prodotti da InceptionV3.
- d è la dimensionalità delle feature prodotte (2048 per InceptionV3).

Questa funzione consente di confrontare le feature delle due distribuzioni senza dover assumere che esse facciano parte di una normale multivariata, consentendo di catturare e misurare feature non lineari.

Questa metrica offre alcuni vantaggi rispetto alla FID:

- Non vengono fatte assunzioni rigurdanti le distribuzioni delle feature
- Non presenta alcun bias statistico dal momento che $SMMD$ viene calcolata direttamente dai campioni a disposizione
- Può essere calcolata anche su dataset di dimensioni ridotte per il punto precedente

Capitolo 2

Esperimenti Numerici

In questo capitolo verrà esposto nel dettaglio il setup utilizzato per gli esperimenti condotti. Verrà data particolare attenzione al dataset utilizzato, ai modelli generativi implementati e addestrati e alla sperimentazione riguardante le metriche di valutazione FID e KID.

2.1 Obiettivi

Come anticipato nell'introduzione di questa tesi, l'obiettivo è quello di fornire una versione delle metriche FID e KID che sia adatta anche a valutare immagini completamente diverse dal dataset ImageNet. Queste metriche infatti sono affidabili per valutare la qualità generativa di modelli addestrati su ImageNet o simili, ma con dataset di immagini che differiscono da ImageNet si osserva che i valori di FID e KID non rispecchiano effettivamente la qualità generativa dei modelli.

2.2 Dataset

Il dataset utilizzato è il dataset Mayo. Il dataset in questione contiene immagini di radiografie di sezioni orizzontali del torace umano di diversi pazienti. Il dataset è suddiviso in due porzioni: una porzione per il training e una per il testing (utilizzata per la validazione di modelli di ricostruzione). Le immagini, essendo radiografie, sono immagini in scala di grigi e hanno risoluzione 512×512. Nella figura 2.1 sono mostrate alcune immagini provenienti dal dataset Mayo.

Figura 2.1: Immagini del dataset Mayo

2.3 Sviluppo dei modelli

2.3.1 Strumenti e tecnologie

Lo sviluppo dei modelli è avvenuto tramite il linguaggio di programmazione *Python*. Le librerie principali di cui è stato fatto uso sono:

- **PyTorch**: è una famosa libreria di deep learning, consente lo sviluppo di reti neurali, ottimizzandone l'allenamento su CPU e GPU.
- **Diffusers**: è una libreria facente parte del framework **Hugging Face**, costruita su Pytorch. Implementa un livello di astrazione maggiore rispetto a PyTorch e consente di sviluppare modelli, anche molto complessi, con maggiore facilità.

2.3.2 Data Preprocessing

Le immagini nel dataset Mayo sono immagini in formato `.png`, questo formato non consente l'elaborazione diretta da parte di una rete neurale. Di conseguenza, si è resa necessaria una semplice elaborazione delle immagini, costituita dalle seguenti operazioni:

- **ridimensionamento**: dalla risoluzione iniziale di 512×512 si è scelto di passare alla risoluzione 256×256 per ridurre il tempo e le risorse hardware necessarie per il training.
- **data augmentation**: per i modelli di dimensioni maggiori si è scelto di introdurre una rotazione casuale delle immagini di training per contrastare l'insorgenza di overfitting.
- **normalizzazione e standardizzazione**: in seguito le immagini vengono convertite in tensori *PyTorch* aventi forma `(1,256,256)` i cui valori vengono normalizzati dall'intervallo $[0, 255]$ all'intervallo $[0, 1]$ ($x = \frac{x - x_{min}}{x_{max} - x_{min}}$). Dopo la normalizzazione

avviene la standardizzazione, in modo da mappare i valori da $[0, 1]$ a $[-1, 1]$ per ottenere una procedura di training più stabile ($x = \frac{x-\mu}{\sigma}$ dove $\mu = \sigma = 0.5$).

2.3.3 Configurazioni dei modelli

I modelli implementati sono di due tipologie:

- modelli basati su **U-Net**
- modelli basati su **U-ViT**

Modelli U-Net

In totale sono stati implementati 4 modelli aventi questa architettura, grazie alla libreria *Diffusers*. La tabella 2.1 mostra riassuntivamente le configurazioni scelte per U-Net.

Due modelli sono caratterizzati da un numero ridotto di parametri e seguono la seguente configurazione:

- **Blocco convolutivo 2D**:

$$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 1, C_{out} = 64, k = 3, s = 1, p = 1$$

- **Blocco convolutivo 2D**:

$$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 64, C_{out} = 128, k = 3, s = 1, p = 1$$

- **Blocco convolutivo 2D**:

$$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 128, C_{out} = 192, k = 3, s = 1, p = 1$$

- **Blocco convolutivo 2D**:

$$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 192, C_{out} = 256, k = 3, s = 1, p = 1$$

Ciascuno dei blocchi è composto da due layer convolutivi. La seconda parte della rete è simmetrica alla prima metà sopra definita; la differenza risiede nell'applicazione di layer max pooling dopo i blocchi convolutivi nella prima metà e di blocchi di convoluzione trasposta dopo i blocchi convolutivi nella seconda metà.

L'altro modello implementa un blocco **convolutivo con attenzione** al posto del blocco convolutivo classico, come penultimo blocco nella prima metà:

$$\text{AttnConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 128, C_{out} = 192, k = 3, s = 1, p = 1$$

Gli altri due modelli si caratterizzano per un maggior numero di parametri. Seguono una struttura analoga a quella sopra mostrata, con la differenza che è stato aggiunto un blocco convolutivo per apportare un miglioramento alle performance generative.

In entrambi i modelli è stato aggiunto il seguente blocco convolutivo:

$$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 256, C_{out} = 512, k = 3, s = 1, p = 1$$

.

	Attenzione	Canali	Profondità	Parametri
U-Net Small	No	(1, 64, 128, 192, 256)	4	20×10^6
U-Net Small Attn	Sì	(1, 64, 128, 192, 256)	4	21×10^6
U-Net Large	No	(1, 64, 128, 192, 256, 512)	5	62×10^6
U-Net Large Attn	Sì	(1, 64, 128, 192, 256, 512)	5	63.5×10^6

Tabella 2.1: Tabella riassuntiva dei modelli U-Net implementati

Modelli U-ViT

Oltre ai modelli basati su U-Net, che sono i più utilizzati per effettuare generazione tramite diffusione, si è voluto implementare due modelli aventi architettura basata su Transformer, simile a quella proposta nell'articolo originale [15]. La tabella 2.2 mostra riassuntivamente le configurazioni scelte per U-ViT.

I due modelli hanno la medesima architettura, ma sono stati progettati con iperparametri differenti in modo da ottenere un modello di dimensione ridotta (circa 44×10^6 parametri) e uno di dimensione maggiore (circa 100×10^6 parametri).

Il primo modello ha la seguente configurazione:

- **Blocco convolutivo 2D** per realizzare il *patching* dell'immagine:

  $$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 1, C_{out} = d = 512, k = 16, s = 16, p = 0$$

  . Si ha $k = s = 16$ perché i *patch* hanno risoluzione 16×16.

- **12 blocchi transformer**, ciascuno di essi costituito da 2 sotto-blocchi:

– **Blocco di attenzione** che implementata il meccanismo di self-attention definito in 1.3.1:

$$\text{AttnBlock}(h, d, d_k) \quad \text{con} \quad h = 8, d = 512, d_k = 64$$

– **Blocco MLP** (*Multilayer Perceptron*) formato da due livelli totalmente connessi:

$$\text{MLP}(4d, d) \quad \text{con} \quad d = 512$$

- **Blocco convolutivo 2D** finale per produrre la predizione del rumore:

$$\text{ConvBlock}(C_{\text{in}}, C_{\text{out}}, k, s, p) \quad \text{con } C_{in} = 1, C_{out} = 1, k = 3, s = 1, p = 1$$

.

dove h è il numero di teste di attenzione, d è la lunghezza degli embedding, d_k è la lunghezza dei vettori che formano *key, query e value* e $4d$ rappresenta la dimensione del livello intermedio implementato nel blocco MLP.

L'altro modello fa uso degli iperparametri sottostanti:

- d : 768
- d_k : 64
- h : 12

.

	Embedding	Teste	Blocchi	Patch	Parametri
U-Vit Small	512	8	12	16×16	46×10^6
U-Vit Large	768	12	12	16×16	104×10^6

Tabella 2.2: Tabella riassuntiva dei modelli U-ViT implementati

2.3.4 Training

Il training dei modelli è stato possibile grazie al servizio di High Performance Computing offerto dal Dipartimento di Informatica - Scienza e Ingegneria. In particolare, tutti i modelli sono stati allenati su una GPU (Graphics Processing Unit) Nvidia L40 che ha permesso di effettuare e velocizzare il training di modelli di dimensioni elevate.

Tutti i modelli sono stati allenati secondo l'algoritmo 1. Le varianze $\beta_t = 1 - \frac{\alpha_t}{\alpha_{t-1}}$ per lo scheduling del noise sono state definite con $\beta_1 = 10^{-4}$ e $\beta_T = 0.02$, dove $T = 1000$ e l'incremento delle variabili β_t avviene linearmente.

I modelli diffusivi durante il training apprendono come effettuare un'operazione di denoising in più passi, quindi il modello deve imparare come effettuare una rimozione del rumore condizionata dal passo t. Questo richiede un allenamento del modello per un numero di epoche considerevole. Per la casistica in esame si è deciso di optare per un numero di epoche pari a 800 su un dataset di training contenente 3306 immagini, facendo uso di batch formati da 16 immagini.

Per favorire la convergenza del modello è stato utilizzato uno scheduling sinusoidale del learning rate con valore massimo $\eta_{max} = 10^{-4}$ e valore minimo $\eta_{min} = 10^{-6}$.

2.3.5 Sampling

Il campionamento delle immagini è stato realizzato tramite l'algoritmo di generazione velocizzata mostrato in 3. L'utilizzo di questo algoritmo ha consentito una generazione molto più veloce e senza perdita di qualità. Nel capitolo successivo saranno mostrati campioni generati attraverso i vari modelli e verrà mostrato che un numero di passi veramente ridotto può permettere una generazione di qualità ottima.

2.4 FID e KID per imaging medico

Come anticipato in precedenza, le metriche KID e FID, nella loro definizione originale, sono inadatte per constatare e misurare la qualità di un modello generativo per immagini mediche. Si mostrerà nel capitolo successivo che, nonostante i sample generati siano molto simili alle immagini del dataset, le definizioni classiche di FID e KID assumono valori molto elevati.

Questo comportamento è dovuto al fatto che InceptionV3 è un modello addestrato per effettuare task di classificazione su ImageNet. I suoi pesi sono quindi in grado di estrarre feature da immagini simili a quelle di ImageNet per poi effettuare una classificazione di queste feature. Con immagini completamente diverse, come quelle del dataset Mayo, le feature estratte non rappresentano un embedding valido; ciò porta a un **calcolo errato e falsato** delle metriche.

Si è deciso quindi di fornire una versione delle metriche KID e FID, in cui il modello InceptionV3 non fosse allenato su ImageNet, ma su un dataset le cui immagini fossero più simili alle immagini del dataset Mayo, utilizzato in questa tesi.

Le versioni delle metriche KID e FID per imaging medico sono state realizzate facendo uso dei pesi di InceptionV3 pubblicati dagli autori del dataset **RadImageNet** [7]. Oltre a definire il dataset, sono state allenate diverse architetture famose su di esso, al fine di fornire dei modelli da cui partire per effettuare un fine-tuning su dataset più piccoli per task mediche specifiche.

Il dataset in questione è costituito da 1.35×10^6 immagini ottenute da 1.31×10^5 pazienti sottoposti a **tomografia computerizzata**, **risonanza magnetica** ed **ecografia** per patologie muscolo-scheletriche, neurologiche, endocrine, addominali, gastrointestinali, polmonari e oncologiche.

2.4.1 Data preprocessing per KID e FID

Per il calcolo delle due metriche le immagini devono essere processate dalla rete InceptionV3, occorre effettuare alcune operazioni preliminari sulle immagini affinché le shape dei tensori combacino. La shape delle immagini accettata da InceptionV3 è `(3, 299, 299)`, questo perché InceptionV3 è stata pensata per l'elaborazione di immagini RGB. Dal momento che le immagini utilizzate per il training dei modelli generativi esposti precedentemente è `(1, 256, 256)` (le immagini generate hanno la medesima shape), per ciascuna è stato effettuato un sovradimensionamento da 256×256 a 299×299, seguito da una concatenazione di tre copie della stessa immagine lungo l'asse dei canali. Il risultato finale è un'immagine avente forma: `(3, 299, 299)`.

2.4.2 Calcolo di KID e FID

Entrambe le metriche sono state calcolate sull'intero dataset Mayo contenente un totale di 3634 immagini, generando per ciascun modello il medesimo numero di immagini. La FID viene calcolata esattamente come nella formulazione mostrata in sezione 1.5.1. Per quanto riguarda la metrica KID, serve definire la lunghezza $m = n$ dei sottoinsiemi di vettori di feature da confrontare e il numero di questi sottoinsiemi. La KID viene quindi calcolata come in sezione 1.5.2, con la formula mostrata che viene applicata per un numero di volte pari al numero di sottoinsiemi. Il valore finale è dato dalla media aritmetica di tutti i valori ottenuti. Per questa sperimentazione, il numero di sottoinsiemi è pari a 100, ciascuno di essi contenente 256 immagini.

Capitolo 3

Risultati

In questo capitolo saranno esposti i risultati degli esperimenti effettuati. In particolare verrà esposto un confronto delle immagini generate dai modelli sviluppati, verranno mostrate le performance dei modelli in termini di tempo di generazione ed infine saranno confrontati i valori della versione classica delle metriche KID e FID con i valori delle versioni alternative delle due metriche, esposte in 2.4.

3.1 Confronto tra i modelli implementati

Nella seguente sezione verrà mostrata un'analisi visiva delle immagini prodotte dai vari modelli. Saranno mostrati vari confronti fra generazioni con modelli differenti e numero di passi di generazione differenti.

3.1.1 Generazione a basso numero di passi

Un modello diffusivo che fa uso di scheduler DDIM permette la generazione di campioni di elevata qualità anche con un ridotto numero di passi di generazione. Si può osservare come le immagini presenti in figura 3.1, per quanto siano state generate utilizzando solamente 10 passi, presentino una qualità ottima.

L'immagine prodotta dal modello U-ViT Small presenta una leggera rotazione verso destra. Questo effetto è legato alla data augmentation introdotta durante il training per aiutare il modello a evitare l'overfitting.

In figura 3.2 sono mostrate immagini generate con 30 passi. Si osserva come la qualità sia migliore rispetto alla generazione con 10 passi per tutti i modelli.

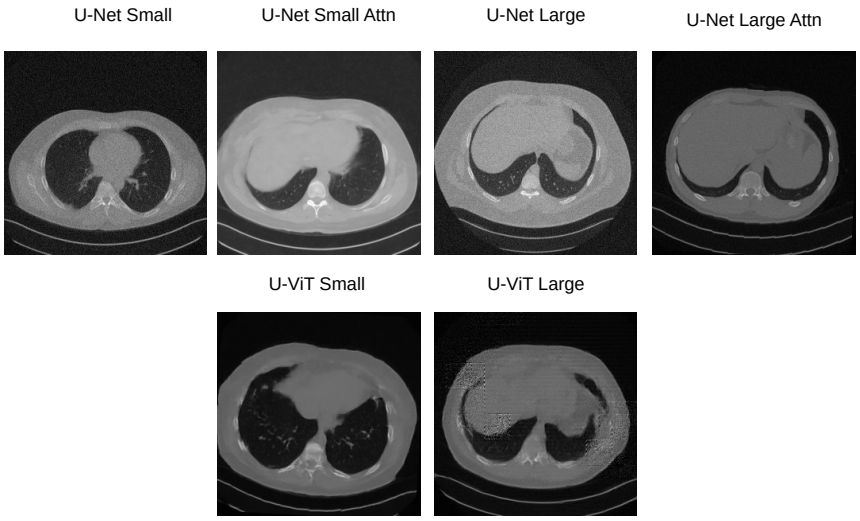

Figura 3.1: Immagini generate con 10 timestep

3.1.2 Generazione a medio numero di passi

Per provare a migliorare ulteriormente la qualità visiva delle immagini si è aumentato il numero dei passi di generazione.

Le immagini generate con 50 passi in figura 3.3 presentano una qualità elevata. Le immagini sono ricche di dettagli e molto simili a quelle del dataset originale, presenti in figura 2.1.

La figura 3.4 mostra immagini ottenute con un processo generativo di 80 timestep. Dagli esperimenti di generazione condotti, con 80 timestep si ha il miglior compromesso tra velocità di generazione e qualità generativa; per questa ragione, per le sperimentazioni su FID e KID descritte in 2.4 e 2.4.2, si è scelto di usare 80 passi per la generazione delle immagini.

3.1.3 Generazione ad alto numero di passi

Fare uso di un numero di step di generazione maggiore di 100 non è necessario se si utilizza uno scheduler DDIM. Tuttavia, aumentando il numero di passi di generazione sono emersi dei comportamenti particolari da parte di uno dei modelli.

Come si può osservare in figura 3.5, con 150 passi la qualità generativa è pressoché la medesima. Andando ad aumentare ulteriormente il numero di passi a 300, tutti i modelli continuano a fornire una generazione di qualità eccellente, ad eccezione di U-ViT Large. Si osserva come con 300 passi generativi (figura 3.6) il modello U-ViT Large non sia in

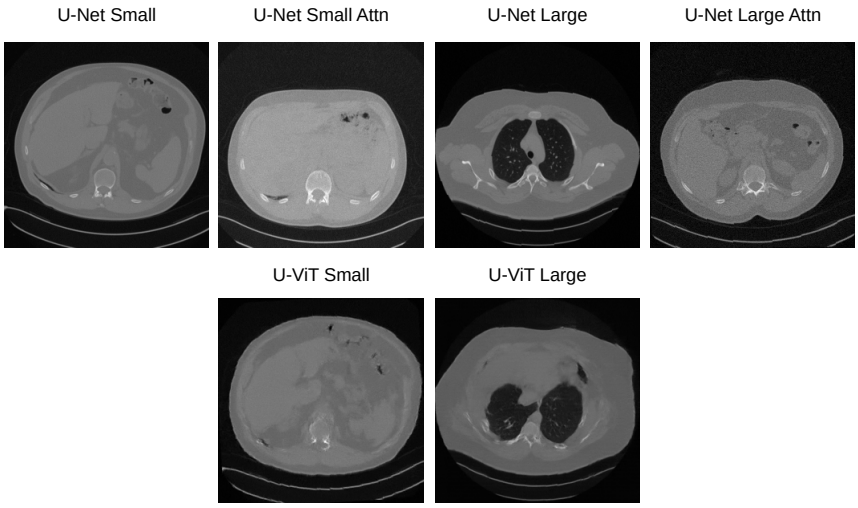

Figura 3.2: Immagini generate con 30 timestep

grado di rimuovere il rumore da diversi patch dell'immagine, fornendo un campione di qualità nettamente inferiore rispetto a quelli degli altri modelli esaminati.

In figura 3.7 sono mostrati altri campioni che rivelano il progressivo deterioramento della qualità generativa del modello U-ViT Large all'aumentare del numero di passi di generazione.

Determinare la causa di questo comportamento anomalo è sicuramente complesso e non è un problema affrontato in questa tesi. Una possibile ipotesi è che il modello U-ViT Large non sia stato allenato per un numero sufficiente di epoche. Ne consegue che il modello effettua delle previsioni leggermente errate nel calcolo del noise. Con un numero medio-basso di passi generativi, la problematica non si nota, ma all'aumentare dei passi, aumenta la quantità di errori commessi dal modello e la propagazione di quest'ultimi, risultando in un peggioramento delle performance generative.

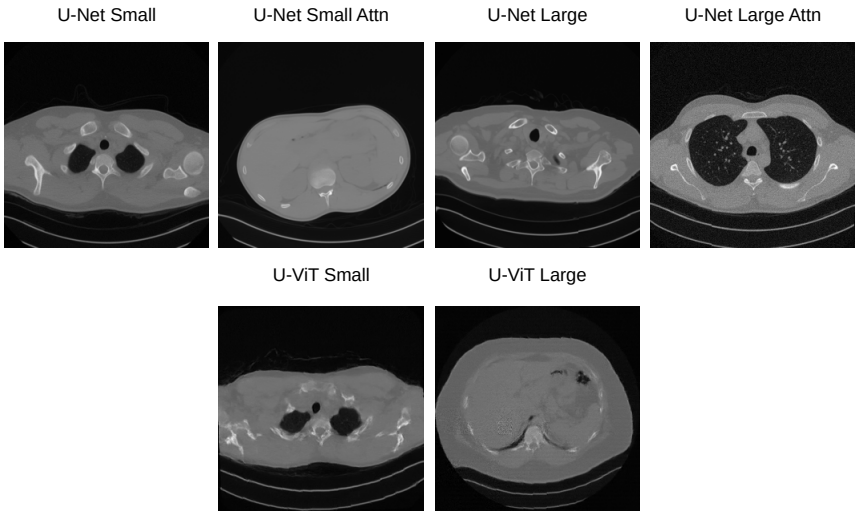

Figura 3.3: Immagini generate con 50 timestep

3.2 Tempi di generazione

In questa sezione sono riportati i risultati dell'analisi dei tempi di generazione dei modelli sviluppati. In particolare è stato misurato il tempo impiegato da ciascun modello per generare un'immagine. Sono state misurate generazioni con i seguenti numeri di passi di generazione: `10, 20, 30, 50, 80, 100, 150, 300, 500, 800, 1000`.

I risultati ottenuti sono riportati nella tabella 3.1.

	10	20	30	50	80	100	150	300	500	800	1000
U-Vit Small	0.066	**0.125**	**0.188**	**0.314**	**0.505**	**0.628**	**0.942**	**1.888**	**3.165**	**5.084**	**6.366**
U-Vit Large	0.068	**0.123**	**0.185**	**0.308**	**0.495**	**0.621**	**0.939**	**1.877**	**3.140**	**5.020**	**6.288**
U-Net Small	0.146	0.248	0.372	0.623	0.994	1.242	1.860	3.713	6.173	9.973	12.366
U-Net Small Attn	**0.175**	0.284	0.424	0.711	1.149	1.428	2.133	4.270	7.149	11.458	14.287
U-Net Large	**0.169**	0.292	0.436	0.733	1.156	1.443	2.190	4.408	7.297	11.658	14.610
U-Net Large Attn	0.227	0.333	0.499	0.840	1.337	1.673	2.520	5.048	8.416	13.457	16.806

Tabella 3.1: Tabella riassuntiva dei dei tempi di generazione

3.2.1 Analisi dei tempi di generazione

La prima osservazione è che il tempo medio di generazione aumenta linearmente all'aumentare del numero dei timestep.

Una seconda osservazione interessante è la velocità elevata dei modelli con architettura U-ViT. Il test di misura dei tempi di generazione dei modelli è stato compiuto eseguendo il sampling su GPU Nvidia L40. Per via dell'architettura altamente efficiente e **pa-**

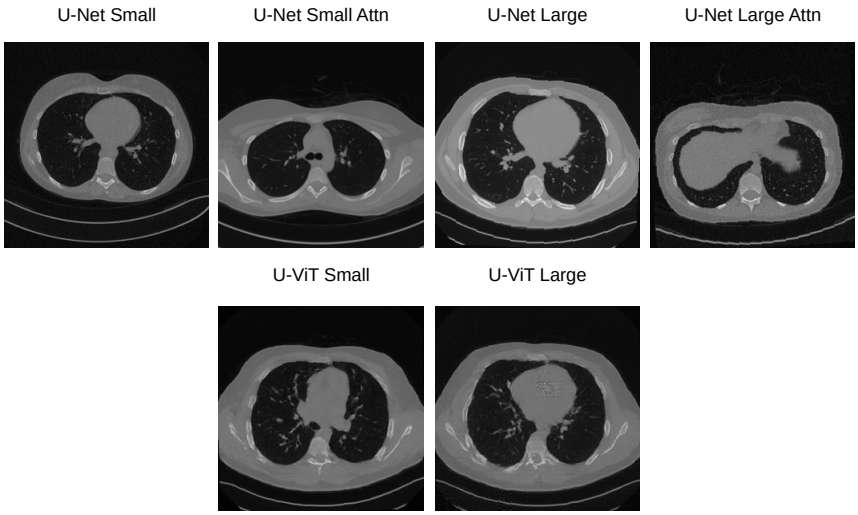

Figura 3.4: Immagini generate con 80 timestep

rallelizzabile di Transformer, l'utilizzo di un hardware ad elevata parallelizzazione ha consentito di raggiungere le elevate velocità di generazione riportate in 3.1.

Si può inoltre notare che, nonostante il modello U-Vit Large abbia più del doppio dei parametri rispetto a U-Vit Small (2.2), le performance sono quasi equivalenti, addirittura leggermente migliori. Anche questo fenomeno è dovuto all'architettura di Transformer, che grazie alla sua **scalabilità** consente di aumentare il numero di parametri dei modelli senza causare un peggioramento delle tempistiche di allenamento ed inferenza.

La quarta e la quinta riga della tabella 3.1 dimostrano che l'utilizzo del meccanismo di attenzione insieme ai blocchi convolutivi aumenta i tempi di generazione. Infatti il modello U-Net Small Attn ha dei tempi di generazione paragonabili a quelli di U-Net Large pur avendo circa un terzo dei parametri .2.1.

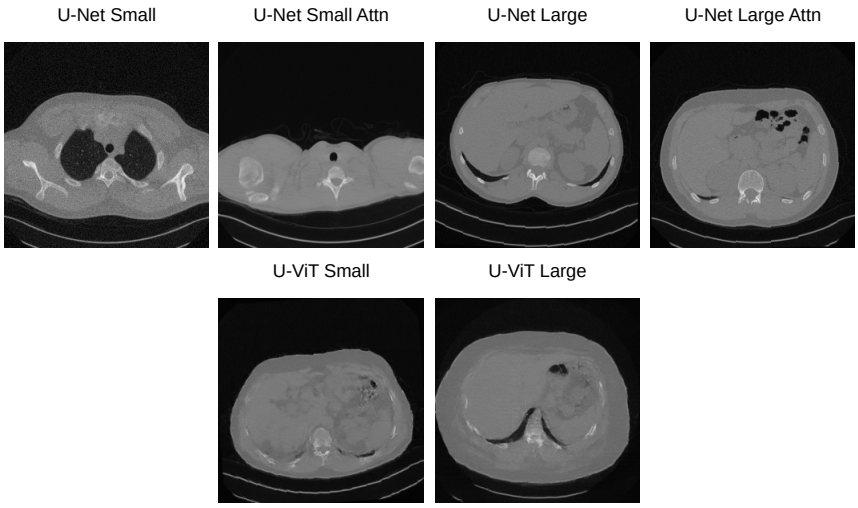

Figura 3.5: Immagini generate con 150 timestep

3.3 KID e FID

L'obiettivo centrale di questa tesi è quello di trovare una soluzione semplice ed efficace che permetta di valutare correttamente la qualità generativa dei modelli diffusivi in ambito medico. In questa sezione saranno mostrati i risultati ottenuti dall'utilizzo delle versioni rivisitate delle metriche KID e FID esposte in 2.4.

Inoltre, grazie a questa nuova formulazione delle metriche, sarà possibile stabilire quale dei modelli è riuscito ad avere le migliori performance nella task di generazione incondizionata sul dataset Mayo.

Le tabelle 3.2 e 3.3 riportano i risultati ottenuti.

Nell'interpretazione dei risultati riportati è importante considerare che la metrica FID andrebbe calcolata su un numero molto alto di immagini. Nell'articolo in cui è stata proposta[4] si consiglia di usare un numero minimo di 10.000 immagini, mentre il numero ideale è di 50.000 immagini. Questi numeri sono possibili solo con dataset di grandi dimensioni, come ImageNet [6], CIFAR-10 o CIFAR-100 [16]. Bisogna riportare che, per via delle ridotte dimensioni del dataset Mayo, le metriche KID e FID sono state calcolate sull'intero dataset (training e testing) in modo da avere il maggior numero di immagini possibile. Va inoltre sottolineato che effettuare il calcolo di queste metriche sui dati di training non è una pratica errata, utilizzata anche dagli autori di DDPM [1].

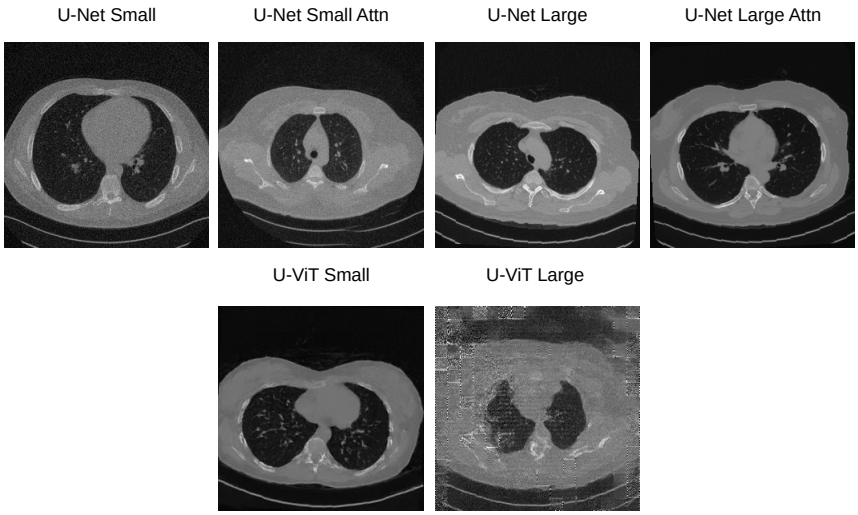

Figura 3.6: Immagini generate con 300 timestep

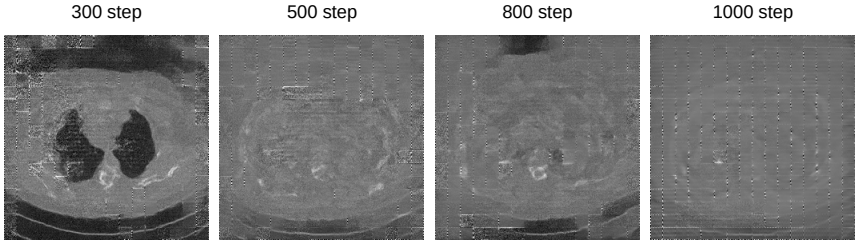

Figura 3.7: Immagini generate con U-ViT Large con timestep elevati

3.3.1 Analisi dei valori di FID e KID

La tabella 3.2 mostra quanto anticipato nell'introduzione di questa tesi: la versione originale delle metriche FID e KID è inadatta a valutare la qualità generativa di modelli allenati su dataset che differiscono di molto da ImageNet. Tutti i valori riportati, difatti, sono molto elevati e non rispecchiano minimamente le capacità generative dei modelli sviluppati (osservabili in sezione 3.1).

La tabella 3.3 conferma la necessità di utilizzare un estrattore di feature adatto per il calcolo di queste metriche. I valori osservati sono molto bassi e confermano la qualità visiva delle immagini osservata in sezione 3.1. I valori delle metriche mostrano anche che i modelli basati su U-Net hanno qualità generativa migliore rispetto ai modelli U-ViT. Il modello U-ViT Large ha performance peggiori del modello U-ViT Small. Questo potrebbe esser dovuto principalmente a due fattori:

- **tempo di allenamento insufficiente**: poiché il modello U-ViT Large ha più di

	FID	KID
U-Vit Small	153.812	0.168
U-Vit Large	145.371	0.146
U-Net Small	75.760	0.054
U-Net Small Attn	122.678	0.049
U-Net Large	48.542	0.039
U-Net Large Attn	63.095	0.048

Tabella 3.2: Valori ottenuti con la versione classica delle metriche FID e KID

	FID	KID
U-Vit Small	1.989	-9.747×10^{-4}
U-Vit Large	6.630	-2.587×10^{-3}
U-Net Small	2.603	2.077×10^{-4}
U-Net Small Attn	3.733	$\mathbf{-8.528 \times 10^{-5}}$
U-Net Large	2.319	-2.704×10^{-4}
U-Net Large Attn	**1.435**	5.357×10^{-4}

Tabella 3.3: Valori ottenuti con la versione di KID e FID basate su InceptionV3 allenata sul dataset RadImageNet

100×10^6 parametri, un training di 800 epoche potrebbe risultare inadatto per un modello di queste dimensioni.

- **dataset di dimensioni ridotte**: i modelli basati su transformer offrono vantaggi in termini di computazione e parallelizzazione, ma richiedono molti più dati rispetto alle reti convoluzionali per ottenere risultati equivalenti. [14]

Capitolo 4

Conclusioni

In questa tesi è stato approfondito l'utilizzo dei modelli diffusivi per la generazione di immagini mediche. I risultati ottenuti dimostrano come questi modelli siano in grado di generare immagini mediche di elevata qualità.
Il confronto tra i modelli implementati ha rivelato che i modelli diffusivi basati su architettura Transformer possono competere con modelli facenti uso di architettura U-Net dal punto di vista della resa generativa, garantendo tempi di campionamento più veloci.
La sperimentazione condotta sulle metriche KID e FID ha permesso di definire delle versioni delle metriche che fossero adatte a confrontare immagini mediche con caratteristiche specifiche e distintive, aprendo alla possibilità di valutare correttamente modelli diffusivi allenati su dataset contenenti immagini molto diverse da quelle presenti in ImageNet.

4.1 Sviluppi futuri

I risultati ottenuti dalle sperimentazioni condotte possono essere ampliati in varie direzioni. Innanzitutto è possibile andare ad apportare modifiche ai modelli diffusivi implementati per tentare di migliorare ulteriormente la resa generativa.

Sarebbe interessante ripetere la sperimentazione sulle metriche FID e KID su dataset differenti e osservare i risultati per consolidare o meno la soluzione proposta in questa tesi.

Le architetture implementate per lo studio realizzato, potrebbero essere utilizzate per le varie applicazioni dei modelli diffusivi in campo medico, tra le quali rientrano: denoising, ricostruzione e super risoluzione di immagini tomografiche o radiografiche. Dai risultati ottenuti si potrebbe realizzare un confronto tra le architetture per verificare se l'elevata resa generativa dei modelli mostrata in questa tesi consente di eseguire in maniera efficace le operazioni sopra nominate.

Riferimenti bibliografici

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[2] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2022.

[3] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," 2021.

[4] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[7] X. Mei, Z. Liu, P. M. Robson, B. Marinelli, M. Huang, A. Doshi, A. Jacobi, C. Cao, K. E. Link, T. Yang, Y. Wang, H. Greenspan, T. Deyer, Z. A. Fayad, and Y. Yang, "Radimagenet: An open radiologic deep learning research dataset for effective transfer learning," *Radiology: Artificial Intelligence*, vol. 0, no. ja, p. e210315, 0.

[8] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," 2016.

[9] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," 2021.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[15] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu, "All are worth words: A vit backbone for diffusion models," 2023.

[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.

Appendice A

Ulteriori generazioni

Per dimostrare le capacità generative dei modelli sviluppati, in questa appendice sono riportate immagini generate dai vari modelli attraverso le quantità di passi seguenti: `10, 30, 50, 80`.

A.1 Generazione con modelli U-Net

A.1.1 U-Net Small

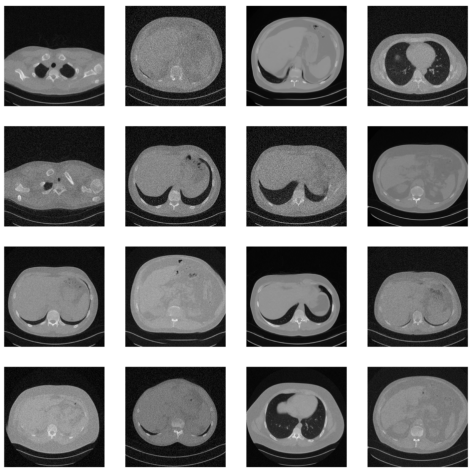

Figura A.1: Immagini generate con U-Net Small: 10 passi di generazione.

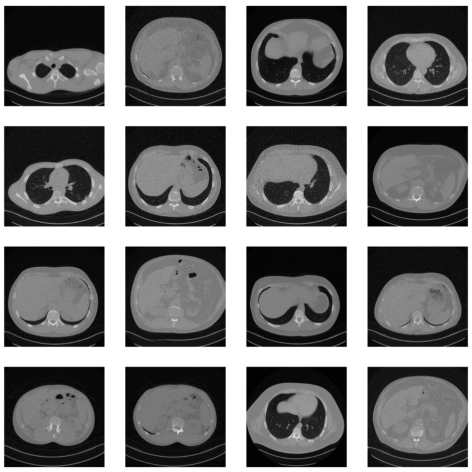

Figura A.2: Immagini generate con U-Net Small: 30 passi di generazione.

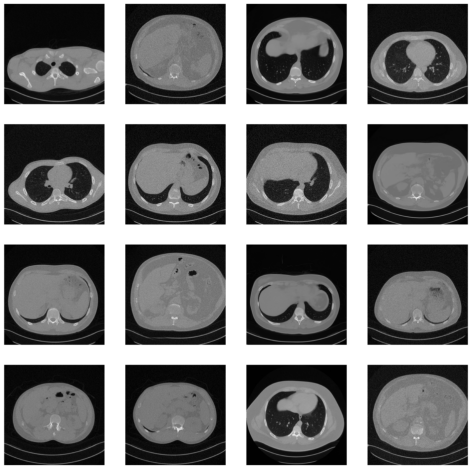

Figura A.3: Immagini generate con U-Net Small: 50 passi di generazione.

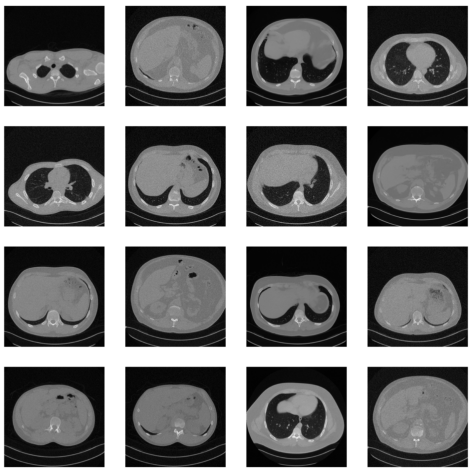

Figura A.4: Immagini generate con U-Net Small: 80 passi di generazione.

A.1.2 U-Net Small Attn

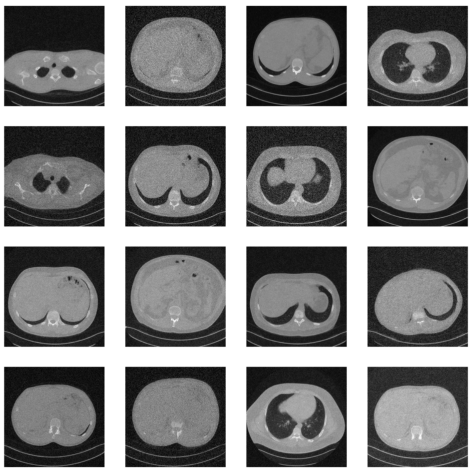

Figura A.5: Immagini generate con U-Net Small Attn: 10 passi di generazione.

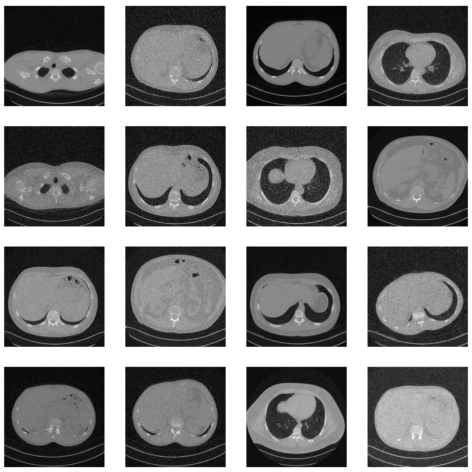

Figura A.6: Immagini generate con U-Net Small Attn: 30 passi di generazione.

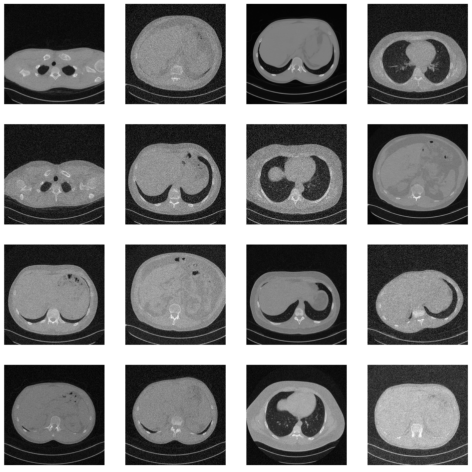

Figura A.7: Immagini generate con U-Net Small Attn: 50 passi di generazione.

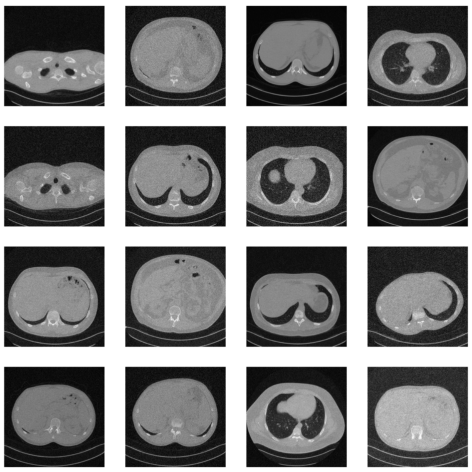

Figura A.8: Immagini generate con U-Net Small Attn: 80 passi di generazione.

A.1.3 U-Net Large

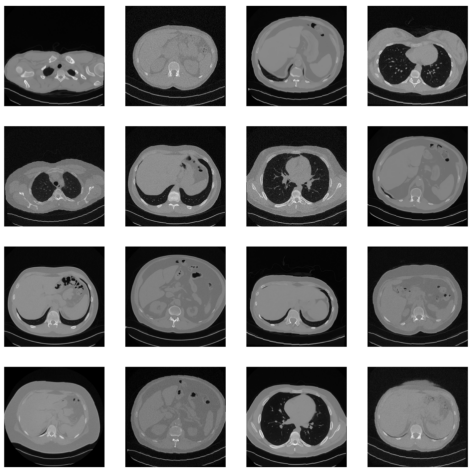

Figura A.9: Immagini generate con U-Net Large: 10 passi di generazione.

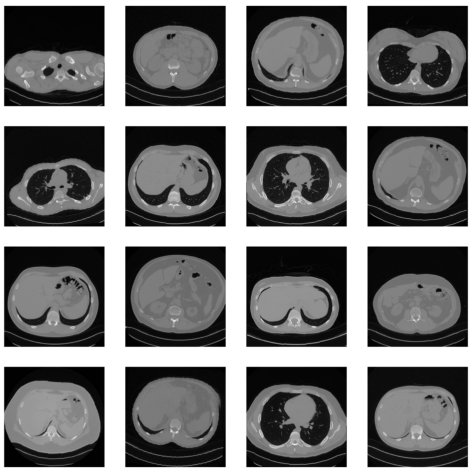

Figura A.10: Immagini generate con U-Net Large: 30 passi di generazione.

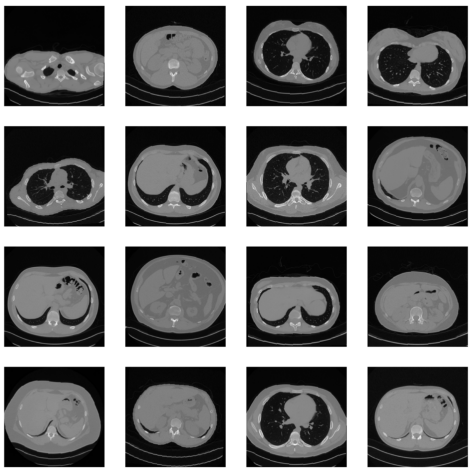

Figura A.11: Immagini generate con U-Net Large: 50 passi di generazione.

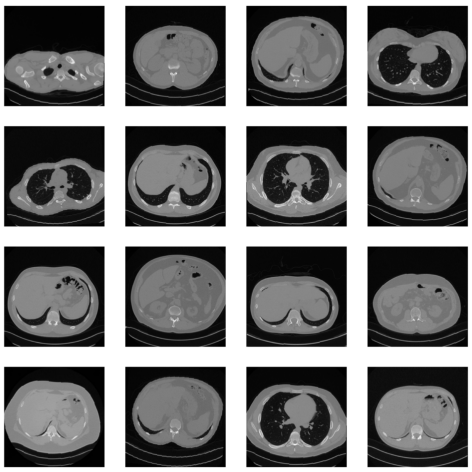

Figura A.12: Immagini generate con U-Net Large: 80 passi di generazione.

A.1.4 U-Net Large Attn

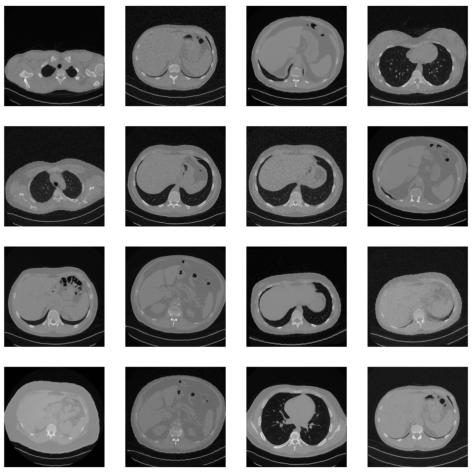

Figura A.13: Immagini generate con U-Net Large Attn: 10 passi di generazione.

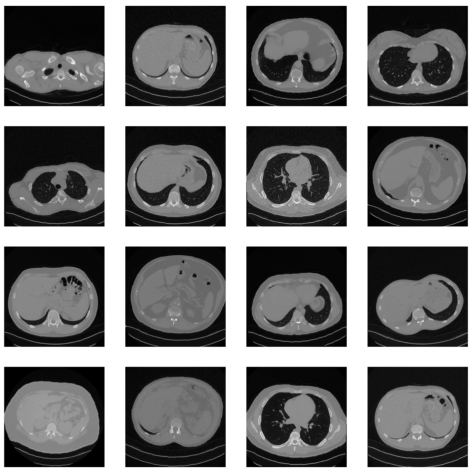

Figura A.14: Immagini generate con U-Net Large Attn: 30 passi di generazione.

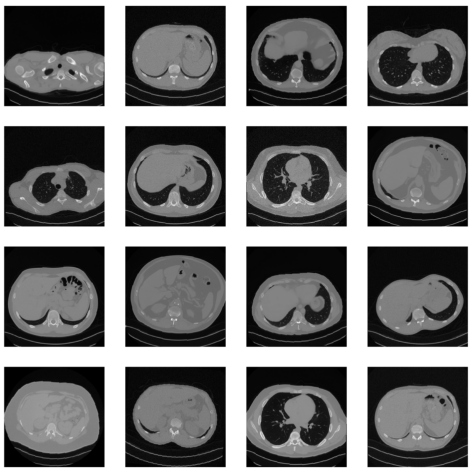

Figura A.15: Immagini generate con U-Net Large Attn: 50 passi di generazione.

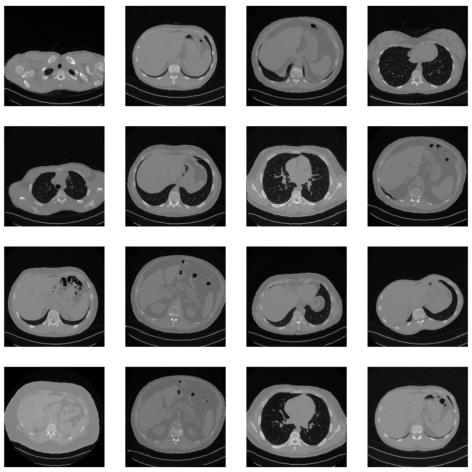

Figura A.16: Immagini generate con U-Net Large Attn: 80 passi di generazione.

A.2 Generazione con modelli U-ViT

A.2.1 U-ViT Small

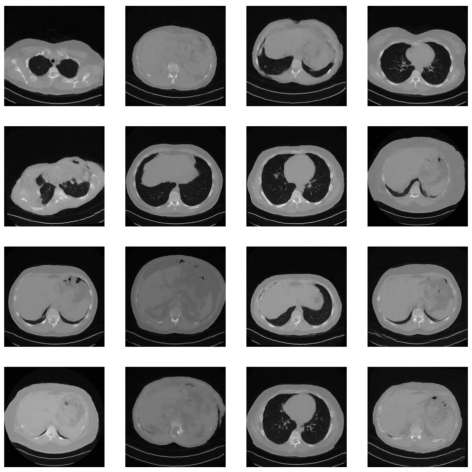

Figura A.17: Immagini generate con U-ViT Small: 10 passi di generazione.

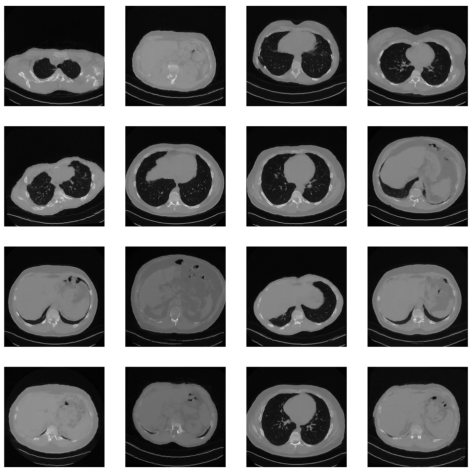

Figura A.18: Immagini generate con U-ViT Small: 30 passi di generazione.

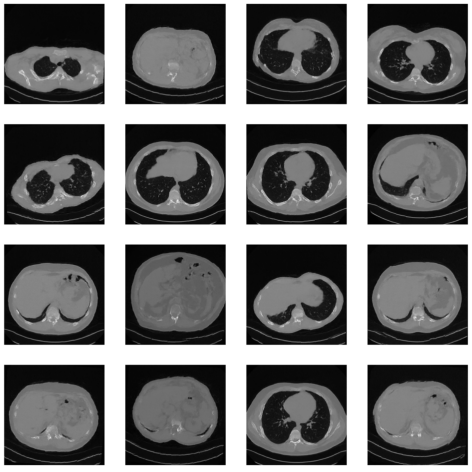

Figura A.19: Immagini generate con U-ViT Small: 50 passi di generazione.

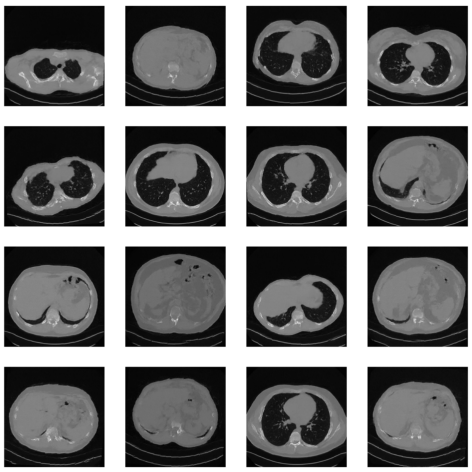

Figura A.20: Immagini generate con U-ViT Small: 80 passi di generazione.

A.2.2 U-ViT Large

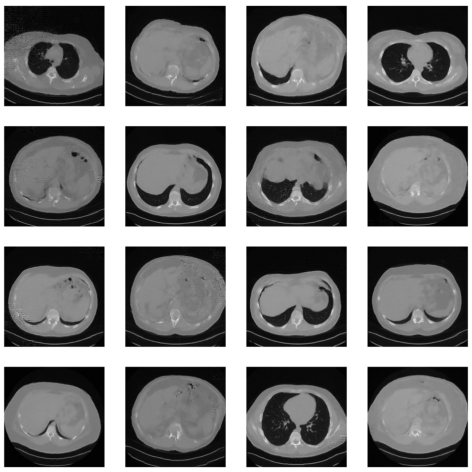

Figura A.21: Immagini generate con U-ViT Large: 10 passi di generazione.

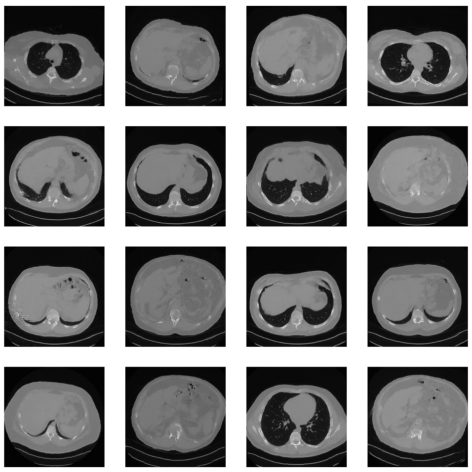

Figura A.22: Immagini generate con U-ViT Large: 30 passi di generazione.

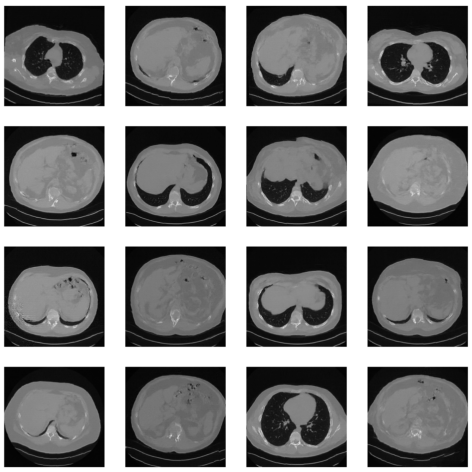

Figura A.23: Immagini generate con U-ViT Large: 50 passi di generazione.

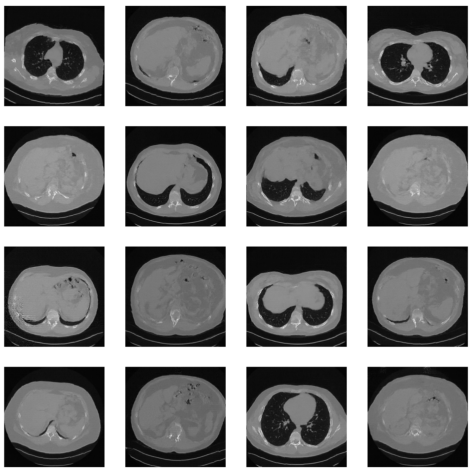

Figura A.24: Immagini generate con U-ViT Large: 80 passi di generazione.

Ringraziamenti

Giunto al termine di questo percorso sento il dovere di esprimere la mia gratitudine verso quelle persone che mi hanno consentito di raggiungere questo importante traguardo.
Ci tengo ad esprimere un sentito ringraziamento al mio relatore, il professore Davide Evangelista. La sua disponibilità e la sua professionalità hanno reso la realizzazione di questo elaborato un'esperienza piacevole e leggera.
Un ringraziamento va alla professoressa Elena Loli Piccolomini, per avermi permesso di esplorare l'entusiasmante mondo della matematica, del calcolo numerico ed in particolar modo dell'imaging medico.
Il ringraziamento più importante va ai miei genitori e a mia sorella, le colonne portanti della mia vita, senza il cui enorme supporto, non avrei mai raggiunto questo incredibile traguardo. Il loro sostegno incondizionato, durante questi tre anni, mi ha spronato a dare sempre il massimo, senza mai sentirmi sotto pressione da parte di essi. Grazie per non aver dubitato di me per nemmeno un momento.
Un ringraziamento speciale va a Sara, fidanzata e migliore amica. La tua capacità di ascoltarmi e di capirmi, la tua abilità nel tranquillizzarmi sono ciò che mi hanno consentito di affrontare i momenti duri di questo percorso. La tua compagnia durante le giornate di studio trascorse insieme ha reso tutto questo ancora più meraviglioso. Questa laurea, in gran parte, è anche tua.
Un grande ringraziamento va a Nicola e Francesco, amici di una vita, per essere sempre stati al mio fianco. Le serate e i momenti passati con voi sono stati una medicina contro preoccupazioni e pensieri negativi.
Ringrazio le ragazze e i ragazzi di Gianluca per tutte le risate e i momenti di spensieratezza trascorsi assieme.
Un ringraziamento va ad Andrea e Luca per aver condiviso insieme a me gli alti e bassi della vita da studenti universitari.
Infine, non posso non ringraziare Oliver. La sua compagnia e il suo affetto incondizionato sono stati essenziali durante questo lungo cammino.