ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Applicazione dei modelli di diffusione all'inpainting e al restauro di immagini

Relatore:
Chiar.mo Prof.
Andrea Asperti

Presentata da:
Beatrice Ravaglia

Sessione II
Anno Accademico 2024/2025

Abstract

La presente tesi si propone di indagare l'impiego della tecnica di *diffusion inpainting* come alternativa ai metodi tradizionali di restauro di immagini basati su U-Net. I risultati ottenuti evidenziano come i modelli di diffusione siano in grado di generare immagini con maggiore nitidezza e livello di dettaglio. Particolare attenzione è stata posta all'uso di strategie di *ensemble*, che hanno permesso di incrementare le prestazioni quantitative sulle metriche MSE, PSNR e SSIM, generando ricostruzioni più stabili.

A supporto di questo lavoro è stato impiegato un *Denoising Diffusion Implicit Model* (DDIM), una variante dei modelli di diffusione ancora poco esplorata in letteratura, che offre significativi vantaggi in termini computazionali grazie alla possibilità di ridurre il numero di step di inferenza senza compromettere la qualità delle ricostruzioni.

La versatilità del metodo è stata confermata dall'applicazione riuscita a un compito alternativo di image reconstruction, ossia la ricostruzione dei canali di colore (*color reconstruction*). In questo scenario, l'integrazione di meccanismi di self-attention nella rete di denoising ha mostrato particolare efficacia, consentendo al modello di cogliere relazioni a lungo raggio tra regioni semanticamente correlate.

Parole chiave: modelli di diffusione, DDIM, inpainting, image restoration, computer vision, deep learning

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione: image restoration

Il **restauro delle immagini** (*image restoration*) consiste in un insieme di tecniche che hanno l'obiettivo di ricostruire una versione quanto più fedele di un'immagine partendo da una sua rappresentazione degradata. Il deterioramento può verificarsi durante la fase di acquisizione, trasmissione o archiviazione e può manifestarsi in diverse forme, come rumore additivo, sfocatura, aliasing o artefatti di compressione.

Esistono varie applicazioni che rientrano nell'ambito dell'image restoration, tra cui la riduzione del rumore (*image denoising)*, la rimozione della sfocatura (*deblurring*), la super-risoluzione, la ricostruzione delle immagini (ad esempio nella tomografia computerizzata), la colorizzazione e il completamento delle parti mancanti (*image inpainting*). La presente tesi si concentrerà in particolare su questi ultimi due punti.

Gli algoritmi di restauro delle immagini hanno assunto nel corso degli anni un'importanza sempre crescente e trovano applicazione in numerosi contesti reali e/o scientifici come l'imaging astronomico, l'elaborazione fotografica, l'imaging medico e molti altri. L'avvento del machine learning ha dato un contributo decisivo al settore, introducendo metodi innovativi e più efficaci per la ricostruzione e il miglioramento di immagini compromesse.

In particolare, tra i vari approcci, i **modelli di diffusione** si distinguono per la loro efficacia nella produzione di immagini realistiche, grazie alla loro capacità di apprendere distribuzioni di dati complesse e di produrre risultati di alta qualità anche in presenza di forti condizioni di degrado.

1.1 Inpainting

L'inpainting è una tecnica di elaborazione delle immagini che consiste nella ricostruzione di parti mancanti o danneggiate. L'obiettivo è quello di ottenere nelle zone da riempire un risultato realistico e visivamente coerente con il contesto circostante, preservando quindi la struttura, la texture e le caratteristiche semantiche dell'immagine originale.

Il concetto di inpainting non è nato in epoca digitale, ma affonda le proprie radici nelle pratiche tradizionali di restauro artistico. Fin dall'antichità, infatti, i restauratori di dipinti, affreschi e manoscritti si sono confrontati con la necessità di ricostruire o integrare parti mancanti di un'opera danneggiata, cercando di riprodurre lo stile e i colori originali così da restituire continuità visiva senza alterare l'armonia complessiva del manufatto [1].

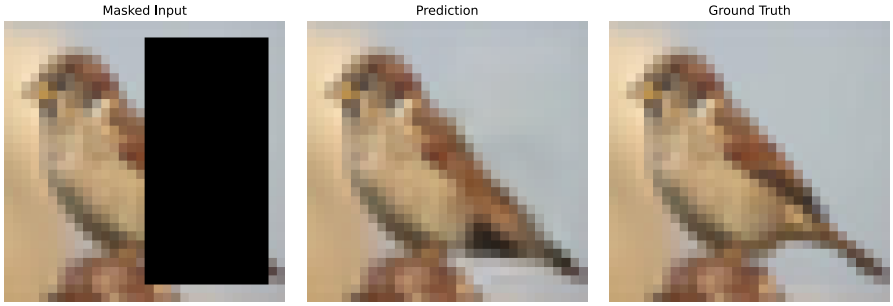

Figura 1.1: Esempio di inpainting su un'immagine del dataset CIFAR-10

Ad oggi, questo campo presenta numerosi ambiti di applicazione, ciascuno di importanza significativa. Si citano alcuni esempi:

- **Restauro di opere d'arte e immagini**: eliminazione di graffi, macchie o parti mancanti senza alterare lo stile originale.
- **Fotoritocco**: rimozione di oggetti indesiderati e correzione di pixel danneggiati o di altri difetti.
- **Video editing**: ricostruzione di frame mancanti o danneggiati, eliminazione di elementi non voluti (es. microfoni, cameraman).
- **Sicurezza e sorveglianza**: ricostruzione di immagini parzialmente oscurate da ostacoli. Supporto nell'analisi forense di immagini o video corrotti o manipolati.
- **Medicina e diagnostica**: ripristino di scansioni mediche con artefatti o parti mancanti, miglioramento della qualità di immagini radiologiche, TAC o risonanze magnetiche.

1.2 Color reconstruction

La *color reconstruction* è il processo di ricostruzione dei valori dei canali di colore mancanti o incompleti di un'immagine, a partire da informazioni parziali sui canali stessi. Seppur rappresenti un problema meno comune rispetto all'inpainting, trova applicazione in diversi ambiti pratici e di ricerca. Ad esempio, è utile nella **compressione e trasmissione di immagini**, in cui alcuni canali possono essere trasmessi in forma ridotta o scomposta, e nell'**elaborazione di immagini multispettrali o satellitari**, dove sensori differenti catturano solo bande specifiche dello spettro e i canali mancanti devono essere ricostruiti.

Figura 1.2: Esempio di ricostruzione dei colori su un'immagine del dataset Oxford Flowers 102

Sebbene il modello di diffusione utilizzato in questa tesi sia stato progettato principalmente per il compito di inpainting, è stato addestrato e valutato anche su questo problema di colorizzazione, al fine di evidenziare la robustezza e la versatilità della tecnica impiegata.

Capitolo 2

Intelligenza artificiale per computer vision

La **computer vision** è il campo dell'informatica che si occupa dell'analisi e dell'elaborazione del contenuto di immagini e video. Tale ambito è stato profondamente trasformato dall'avvento del deep learning: in particolare, le **Convolutional Neural Networks (CNN)** hanno dominato la scena per quasi un decennio, imponendosi come standard de facto a partire dal successo di **AlexNet** nel 2012. Queste architetture hanno portato a significativi progressi in compiti quali classificazione di immagini, rilevamento di oggetti e segmentazione, con modelli noti come **VGG** e **ResNet** che hanno definito i nuovi benchmark del settore [2].

Focalizzandosi sull'inpainting, prima dell'esplosione dei modelli di diffusione, questo tipo di compito veniva generalmente affrontato mediante architetture **U-Net** o altre reti CNN. Tali approcci presentavano limiti significativi, tra cui la tendenza a produrre dettagli sfocati e la difficoltà nel catturare strutture globali complesse, portando talvolta a risultati poco realistici.

Più recentemente, l'introduzione dei **modelli generativi a diffusione** ha portato a miglioramenti rilevanti sia in termini di qualità percettiva (misurata con metriche come FID e LPIPS) sia di fedeltà strutturale (valutata tramite PSNR e SSIM) su dataset standard quali ImageNet, CelebA-HQ e Places2.

Le tecnologie appena citate (quali U-net e modelli di diffusione) verranno analizzate nei prossimi paragrafi; l'obiettivo di questo capitolo è quindi fornire al lettore le conoscenze necessarie per comprendere la tecnologia alla base del lavoro descritto nei capitoli successivi.

2.1 Deep learning e reti neurali

Il **deep learning** è una branca dell'apprendimento automatico (*machine learning*) che trae ispirazione dal funzionamento del cervello umano. Questa metodologia si realizza concretamente attraverso le **reti neurali**, composte da numerosi **neuroni artificiali** organizzati in livelli interconnessi chiamati *layer* [3].

Ogni neurone riceve molteplici input, ne calcola una somma pesata, introduce non linearità tramite una **funzione di attivazione** e produce un singolo output.

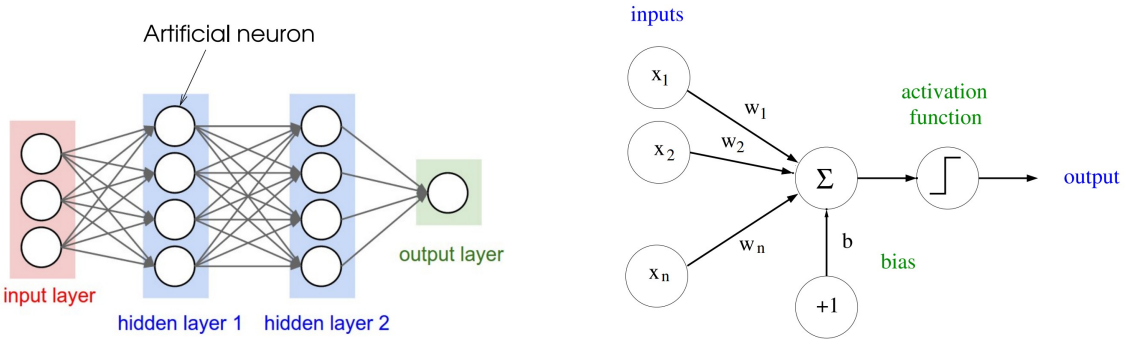

Figura 2.1: Rappresentazione schematica di una rete neurale (a sinistra) e di un neurone artificiale (a destra).

Le reti neurali sono generalmente composte da tre tipi di layer principali, che consentono di simulare la capacità del cervello di generare rappresentazioni astratte a partire da dati grezzi:

- **Strato di input**: riceve i dati grezzi e li trasmette al primo *hidden layer*.
- **Strati nascosti** (*hidden layer*): ciascun livello elabora l'output dello strato precedente e lo passa al successivo. La presenza di molteplici strati nascosti è ciò che permette alle reti neurali di modellare relazioni complesse.
- **Strato di output**: riceve l'ultima rappresentazione e restituisce il risultato finale della rete.

Questa struttura a più livelli permette alle reti neurali di apprendere rappresentazioni gerarchiche del contenuto visivo. Nel contesto dell'inpainting, i primi strati possono catturare dettagli locali come bordi e texture, mentre gli strati più profondi apprendono strutture globali e relazioni semantiche, essenziali per ricostruire in maniera coerente le regioni mancanti dell'immagine.

2.2 U-Net

L'**U-Net** [4] è un'architettura di rete neurale che ha portato a significativi progressi nell'ambito della computer vision. Si tratta di un particolare tipo di **Convolutional Neural Network (CNN)** utilizzata prevalentemente per compiti di segmentazione e ricostruzione di immagini.

Le U-Net sono caratterizzate da due componenti principali:

- **Compressione (*down-scaling*)**: in questa fase la rete riduce progressivamente le dimensioni spaziali dell'immagine di input tramite layer convoluzionali e operazioni di *pooling*. La compressione permette di catturare le caratteristiche più rilevanti, tralasciando dettagli locali. Per compensare questa riduzione, il numero di canali aumenta ad ogni livello: ciascun canale può rappresentare un tipo differente di feature, dalle più semplici, come bordi e texture, a quelle più complesse, come forme e strutture globali.
- **Decompressione (*up-scaling*)**: in questa fase la rete riporta le feature alla risoluzione originale tramite operazioni di upsampling o convoluzioni trasposte. Grazie all'impiego di *skip connections*, le caratteristiche estratte nella fase di compressione vengono riposizionate nei punti corretti.

Il livello centrale dell'architettura è chiamato **collo di bottiglia** (*bottleneck*), e funge da ponte tra compressione ed espansione. In questa fase la rete ha catturato le informazioni più astratte e globali dell'immagine: le dimensioni spaziali sono ridotte al minimo e il numero di canali è massimo.

Questa struttura di tipo encoder-decoder conferisce alla rete la caratteristica forma a "U", come si può osservare nella figura 2.2.

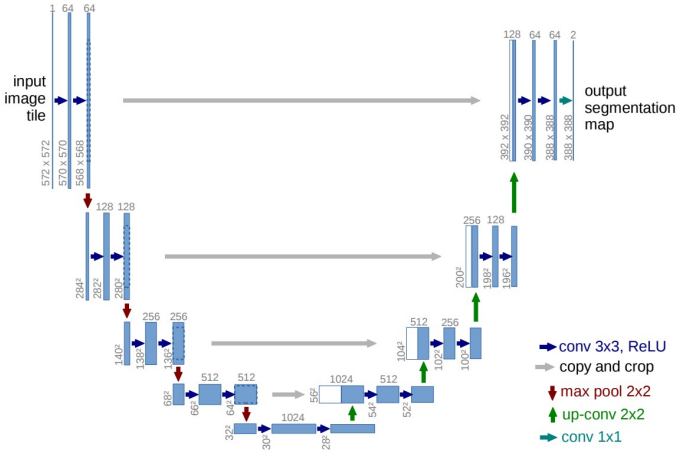

Figura 2.2: Architettura tipica di una U-Net.

2.2.1 Layer convoluzionali

I layer convoluzionali costituiscono il nucleo delle CNN e sono fondamentali per l'estrazione delle caratteristiche dalle immagini. L'operazione alla base di questi strati è la **convoluzione**, che consiste nell'applicare un **filtro o *kernel*** facendolo scorrere sull'immagine. In ogni posizione, il filtro viene sovrapposto a una porzione dell'immagine: i valori corrispondenti vengono moltiplicati elemento per elemento e successivamente sommati, generando un singolo valore della nuova mappa di caratteristiche. Un semplice esempio viene riportato nella figura 2.3

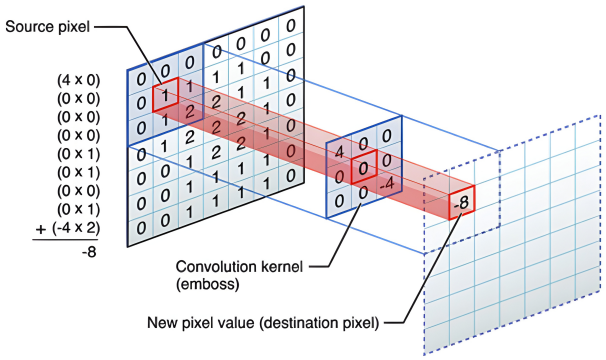

Figura 2.3: Esempio di convoluzione con kernel 3×3.

Il risultato è quindi una nuova mappa di caratteristiche, detta *feature map*, che evidenzia pattern specifici dell'immagine come bordi, texture o altre strutture rilevanti.

Un aspetto chiave dei layer convoluzionali è il concetto di **campo recettivo**: ogni neurone non è influenzato da tutti i neuroni dello strato precedente, ma solo da un piccolo sottoinsieme adiacente. In pratica, ogni neurone agisce come un filtro che si attiva in presenza di un pattern specifico nell'immagine di input. In questo modo, i layer convoluzionali permettono alla rete di costruire una rappresentazione gerarchica dell'immagine, partendo da caratteristiche locali semplici nei primi strati fino a pattern complessi e strutture globali negli strati più profondi [5].

2.2.2 Skip connection

Le *skip connection* costituiscono un elemento cruciale dell'architettura U-Net. La loro funzione principale è quella di collegare direttamente le *feature map* generate durante la codifica con quelle corrispondenti della decodifica, preservando dettagli spaziali che altrimenti verrebbero persi durante le operazioni di convoluzione e *pooling*.

Le skip connection consentono di integrare le rappresentazioni astratte e globali, apprese negli strati più profondi della rete, con i dettagli spaziali fini provenienti

dagli strati più superficiali. Questa capacità di combinare informazioni a diversi livelli di astrazione rende le U-Net particolarmente efficaci in compiti che, come l'inpainting, richiedono accuratezza sia locale che globale.

2.3 Transformer

I **Transformer** sono un'architettura di rete neurale introdotta da un gruppo di ricercatori di Google nel 2017 [6] originariamente per compiti di elaborazione del linguaggio naturale, ma che negli ultimi anni ha trovato applicazioni anche nella computer vision. La loro caratteristica principale è il meccanismo di **self-attention**, che costituisce il cuore del modello e che verrà approfondito nella sottosezione successiva.

Un Transformer è composto da un encoder e un decoder, organizzati in blocchi ripetuti.

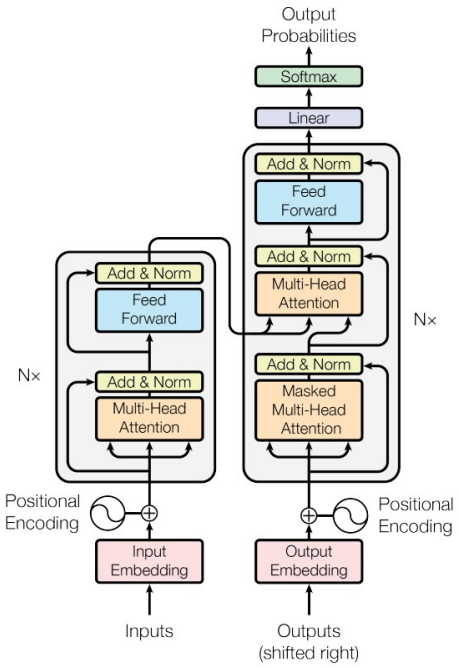

Figura 2.4: Architettura tipica di un Transformer.

2.3.1 Attention

Nel machine learning, il meccanismo di **attention** consente al modello di valutare l'importanza relativa di ciascun elemento dell'input rispetto agli altri. Questo concetto si ispira al funzionamento dell'attenzione umana, che permette di focalizzarsi sugli stimoli più rilevanti in un determinato contesto.

Tale meccanismo è alla base dei Transformer e permette alla rete di concentrarsi selettivamente sulle parti più rilevanti dell'input.

In termini matematici, l'attention può essere definita come:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

dove Q (*query*), K (*key*) e V (*value*) sono matrici derivate dall'input: le *query* rappresentano ciò che il modello sta "cercando" o a cui vuole prestare attenzione, le *key* (con dimensione d_k) fungono da etichette per confrontare le query, mentre i *value* contengono i contenuti informativi veri e propri da propagare agli strati successivi. La funzione *softmax* trasforma i valori in pesi normalizzati la cui somma equivale a 1, indicando l'importanza relativa di ciascun elemento.

La ***self-attention*** è una forma di attention in cui query, key e value provengono dallo stesso input. Questo meccanismo consente di catturare dipendenze a lungo raggio (ad esempio, tra i diversi elementi di un'immagine) senza ricorrere a convoluzioni.

I Transformer utilizzano in particolare la ***multi-head attention***, la quale estende il concetto di self-attention eseguendo più meccanismi di attenzione in parallelo, ciascuno con parametri differenti. Formalmente, se si hanno h teste di attenzione:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.2}$$

dove le diverse teste (*head*) vengono concatenate e poi trasformate da W^O, una matrice di pesi. Ogni testa è definita come:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

W_i^Q, W_i^K, W_i^V e W^O sono matrici di pesi apprese durante l'addestramento. Il vantaggio della *multi-head attention* è quello di permettere alla rete di catturare diversi tipi di relazioni tra gli elementi dell'input contemporaneamente, migliorando l'espressività del modello.

2.4 Modelli generativi

I modelli generativi sono una classe di reti neurali il cui scopo è quello di apprendere la distribuzione p_{data} dei dati reali a partire da un insieme di campioni disponibili (il dataset di training). L'obiettivo principale è costruire una distribuzione p_{model} che

sia il più possibile vicina a p_{data}, in modo da poter generare nuovi campioni realistici e coerenti con il dominio osservato.

2.4.1 Modelli generativi a variabili latenti

Nei modelli generativi a variabili latenti, la probabilità di un dato osservato X viene espressa attraverso variabili non osservabili z, dette **latenti**.

Durante la fase di training, il modello riceve in input i dati reali X e apprende come mapparli in uno spazio latente z e, inversamente, come ricostruire X a partire da z. L'obiettivo è imparare una distribuzione condizionata $P(X|z)$ che sia coerente con i dati osservati. La seguente formula rappresenta la **distribuzione marginale** dei dati che il modello generativo cerca di approssimare:

$$P(X) = \int P(X|z)P(z)\,dz \tag{2.4}$$

dove:

- X sono i dati reali;
- z rappresenta una codifica astratta di X nello spazio latente;
- $P(z)$ è la distribuzione **a priori (*prior*)** sullo spazio latente, scelta in genere semplice e nota, ad esempio una distribuzione gaussiana standard;
- $P(X|z)$ è la distribuzione condizionata, che descrive come generare un dato osservabile X a partire da una variabile latente z.

Il vantaggio di questa formulazione è che, invece di modellare direttamente la distribuzione dei dati $P(X)$, troppo complessa, si suddivide il problema in due parti più semplici: scegliere un *prior* $P(z)$ nello spazio latente e imparare una distribuzione condizionata $P(X|z)$ coerente con i dati reali.

In fase di inferenza (o **campionamento**, *sampling*), invece, non si dispone di X reale: si campiona un vettore latente $z \sim P(z)$ e lo si passa al generatore, che produce un nuovo campione $\hat{X}$. In questo modo il modello è in grado di generare dati realistici che appartengono al dominio appreso durante l'addestramento.

Un concetto chiave è quello del ***manifold* dei dati**, ossia dello spazio delle configurazioni realistiche: il modello deve cercare di generare campioni che appartengano al dominio delle immagini naturali, evitando di produrre configurazioni impossibili o non plausibili.

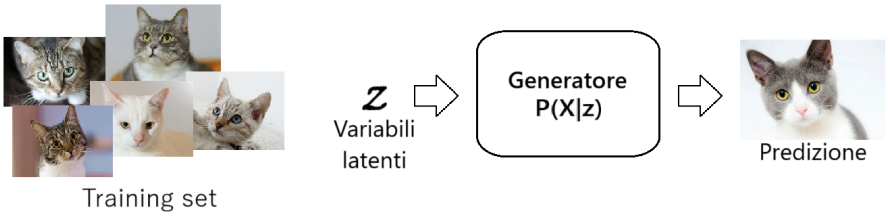

Figura 2.5: Schema di un modello generativo che sfrutta la distribuzione $P(X|z)$ per generare un nuovo campione partendo da $z \sim P(z)$.

.

2.4.2 Tecniche ensemble

Una caratteristica peculiare dei modelli generativi è la loro natura **non deterministica**: a partire dallo stesso input, il modello può produrre output diversi, poiché il processo di generazione coinvolge variabili casuali latenti. Questo aspetto li distingue dalle reti neurali convoluzionali classiche, come le U-Net, che invece sono modelli deterministici: dato lo stesso input e gli stessi pesi, producono sempre lo stesso output.

Nel contesto dell'elaborazione di immagini, questa variabilità può essere sfruttata a vantaggio del modello. Una strategia comune è infatti quella di generare più predizioni della stessa immagine di input e combinarle tra loro, ad esempio calcolandone la media. In questo modo si ottiene un risultato finale più stabile e spesso più accurato.

L'impiego di tecniche di ensemble risulta particolarmente efficace quando la metrica di valutazione è l'**MSE** (*Mean Squared Error*), poiché la media degli output tende a ridurre la varianza e quindi a minimizzare l'errore quadratico medio.

2.5 Modelli di diffusione

In questa tesi l'attenzione sarà focalizzata su una particolare classe di modelli generativi a variabili latenti: i **modelli di diffusione** [7, 8].

Questi modelli trovano ispirazione nella termodinamica del non equilibrio, in cui un sistema evolve lentamente verso uno stato di maggiore entropia. L'idea alla base dei modelli di diffusione è, infatti, di corrompere progressivamente i dati reali aggiungendo rumore gaussiano (*forward process*) e di addestrare una rete neurale a invertire questo processo, ricostruendo passo dopo passo i dati originali (*backward process* o *reverse process*).

A differenza dei modelli generativi a variabili latenti tradizionali, in cui le variabili latenti rappresentano una versione compressa dei dati reali, i modelli di diffusione operano direttamente nello **spazio visibile** dei dati: le versioni rumorose delle immagini hanno le stesse dimensioni di quelle originali. Questo approccio evita semplificazioni eccessive che potrebbero compromettere la qualità di strutture complesse. Di conseguenza, i modelli di diffusione sono in grado di generare immagini con texture e dettagli realistici, una capacità che ne ha determinato il successo nel campo della computer vision.

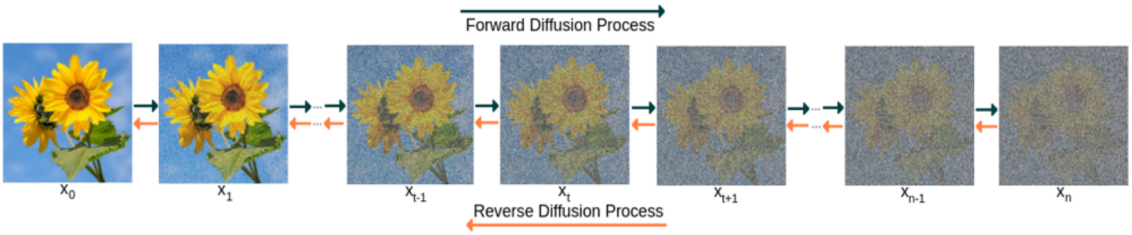

Figura 2.6: Rappresentazione di forward diffusion e reverse process su un'immagine.

.

2.5.1 Il processo di diffusione

Dato un campione x_0 tratto dalla distribuzione reale dei dati $p_{\text{data}}(x)$, il processo ***forward*** consiste nel degradarlo progressivamente aggiungendo piccole quantità casuali di rumore gaussiano in T passi temporali.

A ogni passo $t \in \{1, \ldots, T\}$, il campione x_t è definito come:

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I) \tag{2.5}$$

dove ϵ indica il **rumore gaussiano casuale** e $\{\beta_t\}_{t=1}^{T} \subset (0,1)$ rappresenta la *variance schedule*, ovvero l'intensità del rumore introdotto a ogni passo.

In pratica, la formula 2.5 descrive come, ad ogni passo, il campione precedente venga attenuato (proporzionalmente all'intensità del rumore introdotto) e combinato con rumore gaussiano. Questo processo degrada progressivamente il dato originale, fino a trasformarlo quasi completamente in rumore puro.

2.5.2 Il processo inverso

Il ***backward process*** è il cuore generativo dei modelli di diffusione. Mentre il *forward process* degrada progressivamente i dati aggiungendo rumore, il processo inverso ha lo scopo di ricostruire i dati originali a partire da un campione rumoroso.

Dato un campione x_t contenente una certa quantità di rumore, il modello stima il campione meno rumoroso x_{t-1} mediante una rete neurale addestrata a predire la componente di rumore presente in x_t.

Questo processo è definito come una **catena di Markov parametrizzata**, le cui transizioni hanno la forma:

$$p_\theta(x_{t-1} \mid x_t),$$

dove θ rappresenta i parametri della rete neurale addestrata. Una catena di Markov è un processo stocastico in cui lo stato futuro dipende solo dallo stato attuale e non da quelli precedenti; in questo contesto, significa che la stima di x_{t-1} dipende unicamente dal campione rumoroso x_t e dai parametri della rete, senza considerare direttamente gli stati precedenti.

Ad ogni passo del processo inverso si eseguono due operazioni principali:

1. **Denoising**: rimuovere quanto più rumore possibile presente nel campione corrente, avvicinandosi a un'immagine realistica.
2. **Reiniezione del rumore**: aggiungere nuovamente un piccolo quantitativo di rumore controllato, necessario per mantenere la corretta distribuzione probabilistica del modello.

Questo processo viene ripetuto per tutti i passi $t = T, T-1, \ldots, 1$, fino a ottenere un campione finale $\hat{x}_0$ realistico.

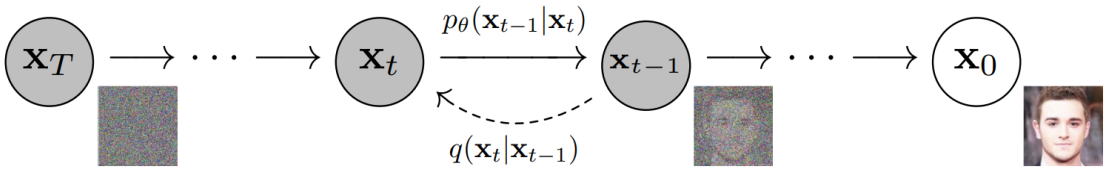

Figura 2.7: Rappresentazione del processo inverso di diffusione.

.

2.5.3 La rete di denoising

Il fulcro dei modelli di diffusione è la **rete di denoising**, ossia una rete neurale (spesso una variante di U-Net) che viene addestrata a predire il rumore ϵ contenuto in un campione rumoroso x_t.

La rete riceve in input:

- il campione rumoroso x_t,

- α_t, che rappresenta la quantità di segnale residua (*signal rate*) nell'immagine rumorosa.

Partendo da questi dati, la rete produce una stima del rumore $\epsilon_\theta(x_t, t)$.

Formalmente, la ricostruzione del campione $\hat{x}_0^t$ può essere stimata come:

$$\hat{x}_0^t = \frac{x_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(x_t, \alpha_t)}{\sqrt{\alpha_t}}. \tag{2.6}$$

Si osservi come la rete di denoising rappresenta l'unico componente del modello che viene effettivamente addestrato.

2.5.4 Tecniche di noise-reinjection: DDPM e DDIM

Esistono due principali modalità per la re-iniezione del rumore durante il *reverse process*:

- Nei **DDPM (*Denoising Diffusion Probabilistic Models*)** ad ogni passo del processo viene aggiunto un nuovo rumore gaussiano campionato casualmente. Questo rende la generazione **stocastica**.
- Nei **DDIM (*Denoising Diffusion Implicit Models*)**, invece, ad essere reiniettato è lo stesso rumore rimosso, ma in forma attenuata. In questo modo il processo diventa, a parità di rumore iniziale, **deterministico**, permettendo anche un campionamento più veloce.

2.6 Diffusion inpainting in letteratura

Negli ultimi anni, i modelli di diffusione hanno ottenuto risultati notevoli nell'ambito dell'inpainting, diventando una delle tecniche più utilizzate nel campo. L'obiettivo di questo paragrafo è di fornire una panoramica delle soluzioni proposte e delle caratteristiche ricorrenti dei principali approcci adottati in letteratura.

I diversi lavori analizzati presentano alcuni elementi in comune: la maggior parte dei modelli utilizzati per l'inpainting si basa su **DDPM**, impiega una **loss MSE** e utilizza reti di tipo **U-Net** per il denoising, talvolta arricchite con meccanismi di **self-attention** (come in [9] e [10]). Inoltre, in diversi casi la generazione del rumore è limitata alle sole aree mascherate ([9], [11]). Nel **Capitolo 4** verrà mostrato come il modello adottato nella presente tesi si discosti da alcune di queste scelte implementative.

Il numero di **diffusion step** impiegati è invece estremamente variabile: in letteratura si trovano configurazioni che vanno da meno di 10 step [12] fino a oltre 20.000

[13], a seconda dell'architettura, del dataset e della strategia di training o inferenza adottata.

Alcuni studi, come [10], propongono l'utilizzo di un modello pre-addestrato in maniera incondizionata (ossia addestrato unicamente a generare immagini senza l'uso di maschere) per svolgere il compito di inpainting. Questa strategia permette di gestire maschere di forma arbitraria mantenendo al contempo buone capacità semantiche.

Altri approcci, invece, hanno introdotto tecniche più sofisticate; un esempio è rappresentato da [14], che impiega un metodo di condizionamento basato sulla SVD che consente di ottenere un inpainting non supervisionato, senza la necessità di un training specifico per ogni nuovo scenario.

I modelli di diffusione trovano applicazione non soltanto nell'inpainting, ma anche in numerosi altri ambiti dell'image restoration, quali la rimozione delle ombre (*shadow removal*), il miglioramento di immagini a bassa luminosità (*low-light enhancement*) e la rimozione della pioggia (*deraining*) [15], così come il deblurring e la super-risoluzione [16]. Una delle caratteristiche più rilevanti di tali modelli è la loro flessibilità, che consente di utilizzare la stessa architettura per diversi compiti di restauro, come evidenziato negli studi [9], [12], [14].

I **modelli DDIM** sono molto meno studiati in letteratura rispetto ai DDPM, nonostante offrano vantaggi significativi a livello computazionale grazie al loro processo di inversione deterministica, che permette di generare campioni in modo più rapido e controllato. Recenti lavori hanno mostrato come i DDIM possano essere efficacemente impiegati per compiti di inpainting; ad esempio, in [17] viene proposto un approccio basato su DDIM che integra un innovativo framework bayesiano, il quale guida il processo di denoising assicurando coerenza visiva tra le regioni rivelate e quelle mancanti dell'immagine. Questo approccio supera le limitazioni dei metodi tradizionali, in cui le aree incomplete vengono spesso trattate in modo "separato" dal contesto circostante, e riduce così le incoerenze tra le diverse porzioni dell'immagine.

Similmente, il lavoro [18] sfrutta i modelli di diffusione impliciti per affrontare il problema del *degradation-blind image restoration*, ossia il restauro di immagini degradate senza alcuna conoscenza a priori sulla natura della degradazione applicata. L'approccio proposto mostra come la struttura invertibile dei DDIM possa essere efficacemente utilizzata non solo per l'inpainting, ma anche per un'ampia gamma di altri compiti di restauro: artefatti di compressione, rumore, sfocature, ecc.

Per un quadro più completo sulle metodologie di diffusion inpainting, con informazioni aggiuntive su modelli, tecniche e dataset impiegati in letteratura, si rimanda alle tabelle contenute nell'**Appendice A**.

Capitolo 3

Strumenti utilizzati

In questo capitolo viene presentata una panoramica dell'ambiente di lavoro utilizzato e delle principali librerie software adottate ai fini di questa tesi.

3.1 Risorse HPC: Leonardo

Leonardo [19] è un supercomputer installato nel 2022 e gestito dal Cineca nel Bologna Technopole. È classificato tra i supercomputer più potenti nella **Top500** globale: si posiziona sesto nel mondo e secondo in Europa.

La macchina raggiunge una potenza di quasi 250 PFlops e dispone di 100 PB di capacità di archiviazione. La partizione Booster comprende 3456 nodi, ciascuno equipaggiato con:

- 1 CPU Intel Xeon 8358 a 32 core, 2,6 GHz
- 512 GB di RAM DDR4 3200 MHz (8×64 GB)
- 2 schede Nvidia HDR da 100 Gb/s ciascuna
- 4 GPU Nvidia Ampere personalizzate con 64 GB di memoria HBM2

Le risorse HPC di Leonardo sono accessibili, previa richiesta, ai ricercatori che necessitano di elevate risorse di calcolo. Per accedervi è necessario registrarsi sul sito del Cineca ed essere abilitati come membri di un progetto di ricerca. Una volta completata la registrazione, è necessario digitare da terminale il seguente comando:

```
step ssh login "[email]" -provisioner cineca-hpc
```

Questo comando apre una pagina web di autenticazione a due fattori, nella quale è necessario inserire email, password e una OTP. Completata la procedura, si può infine accedere a Leonardo mediante:

```
ssh [username]@login.leonardo.cineca.it
```

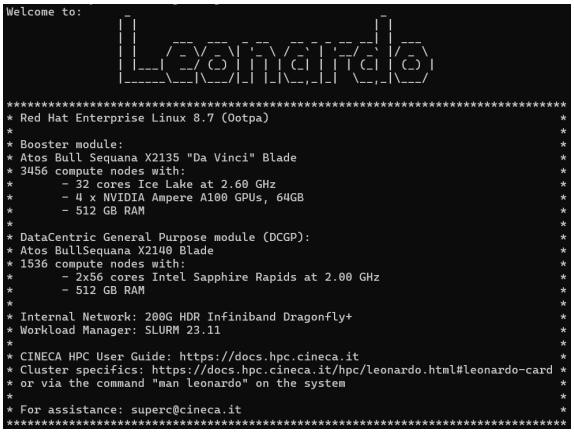

Figura 3.1: Interfaccia a riga di comando del supercomputer Leonardo dopo l'accesso SSH.

.

Prima di eseguire il proprio codice, è necessario caricare i moduli contenenti le librerie necessarie. In questo caso, nello specifico, sono stati utilizzati:

```
module load profile/deeplrn
module load cineca-ai/3.0.0
```

I file possono essere creati o modificati utilizzando un editor di testo come `vi` o `nano`, oppure tramite Visual Studio Code, installando l'estensione "Remote - SSH".

Per eseguire un file sfruttando la potenza di calcolo di `Leonardo`, è necessario creare uno script `sbatch` con il seguente formato:

```
#!/bin/bash
#SBATCH -A [nome_progetto]
#SBATCH -p boost_usr_prod
#SBATCH --time [tempo:limite:esecuzione]   # formato: HH:MM:SS
#SBATCH -N 1                               # numero di nodi
#SBATCH --ntasks-per-node=4                # tasks per nodo
#SBATCH --gres=gpu:4                        # GPU per nodo
#SBATCH --mem=123000                        # memoria per nodo in MB
#SBATCH --job-name=my_batch_job

python3 ./[nome_file.py]
```

Lo script viene eseguito tramite il comando `sbatch [nome_script]`. Questo crea un job a cui viene assegnato un numero progressivo in base alla coda degli altri job in esecuzione.

È possibile inoltre controllare lo stato del job (ad esempio `PD` = pending, `R` = running) tramite `squeue -j [numero_job]`.

L'output generato dal job viene salvato in un file chiamato `slurm-[numero_job].out` nella cartella in cui è stato eseguito.

Per interrompere un job o rimuoverlo dalla coda, si utilizza `scancel [numero_job]`.

3.2 Librerie Python

Nel contesto del machine learning, il linguaggio di programmazione **Python** si distingue per la sua diffusione e versatilità, mettendo a disposizione una vasta gamma di librerie dedicate alla manipolazione dei dati e allo sviluppo di reti neurali.

Si riporta di seguito una breve presentazione delle principali librerie impiegate.

3.2.1 TensorFlow

`TensorFlow` è una libreria open source sviluppata da Google per il calcolo numerico e l'implementazione di reti neurali. Permette di costruire, addestrare e valutare modelli di Machine Learning in modo efficiente su CPU e GPU.

3.2.2 Keras

`Keras` è un'API di alto livello per reti neurali integrata in TensorFlow che semplifica la creazione e l'addestramento di modelli complessi. Permette inoltre di definire modelli personalizzati estendendo la classe `keras.Model` e di utilizzare *callback* per monitorare metriche, salvare checkpoint dei modelli e gestire *early stopping* durante l'addestramento.

3.2.3 NumPy

`NumPy` è una libreria fondamentale per il calcolo scientifico in Python. Fornisce array multidimensionali, strutture dati e funzioni matematiche e statistiche per la manipolazione dei dati. È essenziale sia per il preprocessamento dei dati prima dell'addestramento dei modelli, sia per eseguire operazioni di verifica e analisi sui risultati ottenuti. Viene spesso usato in combinazione con altre librerie scientifiche come `Pandas` e `SciPy` per la gestione di dataset complessi e per calcoli più avanzati.

3.2.4 Matplotlib

`Matplotlib` è una libreria Python per la visualizzazione di dati tramite grafici e figure 2D personalizzabili. La maggior parte delle figure presenti in questa tesi per rappresentare l'andamento delle metriche e la visualizzazione delle immagini processate è stata realizzata tramite questa libreria

Capitolo 4

Implementazione e dettagli architetturali

In questo capitolo vengono presentati i dettagli architetturali del modello di diffusione utilizzato, originariamente progettato dal professor Asperti e dai suoi collaboratori, e successivamente adattato dalla sottoscritta per adeguarlo ai dataset utilizzati e ai compiti specifici della tesi. Il modello era inizialmente concepito per il compito di ricostruzione della temperatura superficiale marina; nel contesto di questa tesi, l'obiettivo è stato verificare la robustezza della tecnica testandone l'efficacia nell'**inpainting su dataset differenti**, in particolare **EuroSAT/RGB**. È stata inoltre valutata la versatilità del modello mediante un compito differente di image restoration: la **ricostruzione dei colori** (*color reconstruction*) sul dataset **Oxford Flowers 102**. Le prestazioni ottenute sono state confrontate con quelle di due U-Net tradizionali e saranno discusse nel dettaglio nel capitolo successivo.

4.1 Dataset e preprocessing

Il preprocessing dei dati è un passaggio fondamentale, poiché consente di migliorare la qualità delle informazioni disponibili, rendere omogenei i formati, normalizzare i valori e preparare i dati in modo che siano adatti all'addestramento dei modelli. In questa sezione vengono descritti i dataset utilizzati in questa tesi e le operazioni di preprocessing applicate.

4.1.1 Inpainting: EuroSAT/RGB

Il dataset utilizzato per il compito di inpainting è **EuroSAT/RGB**. Il set di dati EuroSAT si basa sulle immagini satellitari di Sentinel-2, una missione sviluppata

dall'Agenzia Spaziale Europea nell'ambito del programma Copernicus, ed è costituito da un totale di 27.000 campioni suddivisi in 10 classi. La versione RGB del set contiene le bande di frequenza ottiche R, G, B codificate come immagini con risoluzione di 64×64 pixel.

Il dataset è stato suddiviso in un **set di training**, composto da 20.000 immagini, e in un **set di test**, costituito da 7.000 immagini. Successivamente, è stata applicata una normalizzazione di tipo gaussiano, che consente di trasformare i valori dei pixel affinché abbiano una distribuzione con media pari a zero e deviazione standard unitaria. In questo modo si uniforma la scala dei dati, riducendo l'influenza di valori estremi e facilitando la convergenza durante l'addestramento del modello.

La **normalizzazione gaussiana** può essere espressa matematicamente come:

$$x'_c = \frac{x_c - \mu_c}{\sigma_c}$$

dove x_c è il valore originale del pixel nel canale c, μ_c è la media dei pixel del training set calcolata sullo stesso canale e σ_c è la deviazione standard. La normalizzazione avviene dunque in maniera indipendente per ciascuno dei canali R, G e B.

Per introdurre la sfida dell'inpainting, al dataset normalizzato è stata applicata una **maschera binaria** che oscura strisce verticali casuali delle immagini: i pixel della maschera con valore 1 indicano zone visibili, quelli con valore 0 indicano le zone oscurate. Le linee verticali hanno spessore variabile e sono posizionate in maniera casuale, in modo che la porzione totale da ricostruire corrisponda a circa il 60–70% della superficie dell'immagine.

Figura 4.1: Inpainting: immagine originale accanto alla propria versione mascherata.

Questa procedura è stata applicata sia al training set sia al test set, permettendo al modello di diffusione di imparare a ricostruire le parti mancanti a partire dalle informazioni residue.

4.1.2 Color reconstruction: Oxford Flowers 102

Per quanto riguarda la color reconstruction, il dataset utilizzato è Oxford Flowers 102, che consiste in 102 categorie di fiori, ciascuna composta da 40 a 258 immagini.

Le immagini sono state fissate alla risoluzione di 128x128 pixel, dopodiché sono state suddivise in un **set di training**, composto da 6.149 immagini, e in due set separati di **validazione** e **test**, ciascuno con 1.020 immagini. Il set di validazione è stato utilizzato per monitorare l'andamento delle prestazioni del modello durante il training.

Per il set di training, considerato relativamente piccolo rispetto alla complessità del modello, è stata applicata la tecnica di ***data augmentation*** al fine di aumentare la quantità e la varietà delle immagini disponibili e migliorare così la generalizzazione. Ogni immagine del set di training è stata generata in sei varianti mediante le seguenti trasformazioni geometriche:

- Originale – immagine invariata.
- Flip orizzontale – immagine specchiata.
- Traslazione a destra – shift orizzontale dei pixel verso destra.
- Flip + traslazione a destra.
- Traslazione a sinistra – shift orizzontale dei pixel verso sinistra.
- Flip + traslazione a sinistra.

Anche in questo caso è stata applicata la normalizzazione gaussiana su ogni canale delle immagini.

Per simulare il compito di color reconstruction è stata applicata una **maschera binaria a tre canali** che divide ogni immagine in tre bande orizzontali di dimensioni simili: la prima conserva solo il canale rosso, la seconda solo il verde e la terza solo il blu, azzerando in ciascuna banda i canali rimanenti.

Figura 4.2: Colorizzazione: immagine originale accanto alla propria versione mascherata.

.

Questa maschera è stata applicata a tutti i set (training, validazione e test), permettendo di addestrare e valutare la capacità del modello di completare i canali mancanti e ricostruire correttamente i colori.

4.2 Reti di denoising

Come già evidenziato in precedenza, la rete di denoising costituisce il fulcro dei modelli di diffusione ed è, di fatto, l'unico componente che può essere addestrato. Nel lavoro svolto per questa tesi sono state sperimentate diverse varianti della rete di denoising; nei prossimi paragrafi verranno presentate le due versioni che hanno mostrato le prestazioni migliori.

È importante precisare che, mentre nei modelli di diffusione tradizionali la rete di denoising stima generalmente il rumore presente nell'immagine, in questo lavoro la rete predice direttamente il **segnale utile**, ossia l'immagine finale da ricostruire. Il rumore può essere comunque ricavato a posteriori tramite la formula:

$$\hat{\epsilon} = \frac{x_t - \alpha_t \, \hat{x}_0}{\sigma_t}, \tag{4.1}$$

dove x_t è l'immagine rumorosa al passo t, $\hat{x}_0$ è l'immagine predetta dalla rete, α_t rappresenta il coefficiente di segnale (*signal rate*) e σ_t il coefficiente di rumore (*noise rate*).

4.2.1 Residual U-Net

Tra le architetture di denoising sperimentate, ad essersi dimostrata particolarmente efficace è stata una **U-Net residuale** (*Residual U-Net*).

L'architettura adotta la struttura tipica encoder–decoder con *skip connection*, arricchita da due elementi distintivi, che verranno successivamente analizzati nel dettaglio:

- l'utilizzo di **blocchi residuali** (*residual blocks*);
- l'inclusione di un **embedding sinusoidale** che rappresenta il livello di rumore.

Tale U-Net ha una profondità complessiva di 3 livelli, con un numero di canali crescente pari a **64, 128 e 256**.

La fase di **downsampling** comprende due blocchi con 64 e 128 canali, che riducono progressivamente la risoluzione spaziale dell'immagine mediante **convoluzioni** e operazioni di ***average pooling***. Il livello più profondo, il **collo di bottiglia**, utilizza 256 canali e serve a elaborare le rappresentazioni a risoluzione ridotta prima della ricostruzione.

La fase di **upsampling** comprende due blocchi con 128 e 64 canali, che ricostruiscono l'immagine originale aumentando la risoluzione tramite **interpolazione bilineare**,

una tecnica che stima i valori dei pixel mancanti calcolando una media ponderata dei pixel vicini.

Infine, un layer convoluzionale con **kernel 1×1**, inizializzato a zero, produce l'output finale della rete. Questo layer combina linearmente le feature ricostruite dai livelli precedenti e riduce il numero di canali all'output desiderato, corrispondente all'immagine predetta.

Il totale dei parametri, come mostrato nella tabella 4.1, ammonta a circa 6M, un numero relativamente contenuto per una U-Net residuale.

Tabella 4.1: Numero di parametri della Residual U-Net

Tipo di parametro	**Numero**
Parametri allenabili	6,354,371
Parametri non allenabili	5,120
Parametri totali	6,359,491

4.2.1.1 Residual learning

Il residual learning è un concetto centrale nell'apprendimento automatico, introdotto con le **Residual Networks (ResNet)** e particolarmente utile per l'addestramento di reti neurali profonde. L'idea di base consiste nel semplificare il processo di apprendimento mediante l'aggiunta di connessioni residue che consentono di "saltare" uno o più strati della rete.

In una rete tradizionale, ogni strato cerca di approssimare una funzione obiettivo $H(x)$, che trasforma l'input x nell'output desiderato. Nel residual learning, invece, la rete non apprende direttamente $H(x)$, bensì il residuo rispetto all'input:

$$F(x) = H(x) - x.$$

In altre parole, il modello impara solo la parte "aggiuntiva" $F(x)$ necessaria per trasformare x in $H(x)$. La funzione obiettivo può così essere riscritta come:

$$H(x) = F(x) + x.$$

Questa formulazione introduce un percorso di identità, ovvero una connessione diretta che bypassa alcuni strati e permette all'input x di essere trasmesso intatto [20].

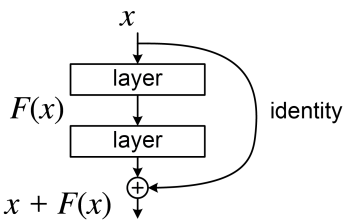

Figura 4.3: Schema di un blocco residuale che consente di saltare due layer.

.

Nella U-Net residuale utilzzata sono presenti **3 residual block consecutivi**:

- in ogni blocco di downsampling, prima del *pooling*;
- nel collo di bottiglia;
- in ogni blocco di upsampling, dopo la concatenazione con le *skip connection*.

Questa configurazione migliora la stabilità dell'addestramento e consente predizioni più accurate, poiché la rete apprende a modellare solo la parte aggiuntiva necessaria per correggere l'input, lasciando intatte le informazioni già corrette.

4.2.1.2 Embedding sinusoidale del rumore

Un elemento chiave nella U-Net utilizzata è l'**embedding sinusoidale del livello di rumore**. Questo meccanismo trasforma lo scalare che rappresenta l'intensità del rumore al passo t in un vettore più ricco e multidimensionale, utilizzando funzioni sinusoidali e cosinusoidali a diverse frequenze.

L'obiettivo principale è fornire alla rete un'**informazione temporale continua sul passo corrente** t: il modello deve infatti "sapere" a quale passo del processo di diffusione si trova, in modo da capire il livello del rumore presente in x_t e modulare di conseguenza la predizione dell'immagine pulita.

Nella U-Net residuale utilizzata, l'embedding sinusoidale viene generato a partire da un'informazione sulla quantità di rumore presente (fornita in input alla rete) e successivamente replicato su tutta la dimensione spaziale dell'immagine tramite un'operazione di *upsampling* per essere poi concatenato alle feature map dell'input, così da condizionare l'intera rete sullo stato corrente del processo di diffusione.

4.2.2 Attention U-Net

Un'altra architettura che ha fornito risultati soddisfacenti è la cosiddetta **Attention U-Net**. Questa rete conserva la struttura encoder–decoder con skip connection della U-Net residuale, ma introduce alcune novità, tipiche dei modelli **Transformer**:

- **Self-attention *multi-head*:** nel collo di bottiglia viene applicato un meccanismo di attention, che permette alla rete di catturare meglio relazioni a lungo raggio tra i vari elementi dell'immagine.
- **MLP *Feed-Forward*:** ogni blocco di self-attention include una rete feed-forward (detta ***Multi-Layer Perceptron*, MLP**) che elabora le feature generate dall'attenzione, analogamente a quanto avviene nei Transformer classici.
- **Layer normalization:** ogni blocco convoluzionale residuale è preceduto da una normalizzazione a livello di layer. A differenza della Batch Normalization usata nelle U-Net standard, che normalizza i valori delle feature calcolando media e deviazione standard su tutto il batch, la Layer Normalization normalizza ciascun esempio individualmente. Questo la rende particolarmente adatta ai modelli con meccanismi di attenzione.

La fase di downsampling e upsampling segue la stessa logica della U-Net residuale, con blocchi residuali consecutivi e operazioni di pooling e interpolazione bilineare. Il collo di bottiglia combina residual block e self-attention, fornendo un'elaborazione più ricca delle feature a bassa risoluzione prima della ricostruzione finale. Inoltre, anche in questa rete viene mantenuto il meccanismo dell'embedding sinusoidale del rumore.

Come si osserva nella tabella 4.2, il numero di parametri di questa U-Net è moderatamente più elevato rispetto alla precedente.

Tabella 4.2: Numero di parametri della Attention U-Net

Tipo di parametro	Numero
Parametri allenabili	8,728,771
Parametri non allenabili	0
Parametri totali	8,728,771

4.3 Architettura del modello di diffusione

In questa sezione vengono descritti i dettagli architetturali del modello di diffusione impiegato nel presente lavoro. Il modello consiste in una classe derivata da `keras.Model`, permettendo di definire logiche personalizzate per il training e il processo di diffusione.

4.3.1 Generatore dei dati

Il generatore dei dati impiegato durante il training ha il compito di produrre **batch di immagini** (sottoinsiemi del set di training) pronti per l'addestramento. Per ciascun batch, il generatore concatena lungo l'ultima dimensione (quella dei canali) l'**immagine mascherata**, la **maschera binaria** corrispondente e l'**immagine originale** (*ground truth*) normalizzata, generando un tensore di input di dimensione (B, H, W, C), dove B indica la dimensione del batch, e H e W rappresentano rispettivamente l'altezza e la larghezza delle immagini, mentre C è il numero totale dei canali.

4.3.2 Passo di addestramento

Il passo di addestramento del modello può essere descritto nei seguenti passaggi principali.

Per ogni batch, vengono innanzitutto generate immagini rumorose a partire dalle immagini target (le immagini originali o *ground truth*). Il rumore viene applicato esclusivamente ai pixel da ricostruire, lasciando invariati quelli già visibili.

L'intensità del rumore aggiunto è modulata da due coefficienti, chiamati *noise rate* e *signal rate*, ottenuti nel seguente modo: ciascuna immagine viene associata a un valore casuale t compreso tra 0 e 1, che rappresenta il passo corrente nel processo di diffusione. Dopodiché vengono calcolati:

- ***signal rate*** $= \cos(\theta)$, che indica la frazione di segnale originale ancora presente nell'immagine,
- ***noise rate*** $= \sin(\theta)$, che indica la quantità di rumore introdotta,

dove θ è un angolo calcolato a partire da t. L'uso di angoli garantisce che i due tassi siano complementari, poiché vale la relazione trigonometrica $\sin^2(\theta) + \cos^2(\theta) = 1$. Questo processo prende il nome di *diffusion schedule*.

In questo modo, la rete riceve in input immagini con livelli di rumore variabili, simulando diversi stadi del processo di diffusione. In altre parole, a differenza del-

l'approccio tradizionale, non si iterano tutti i passi della catena di diffusione; invece, il modello impara a effettuare il *denoising* a partire da qualsiasi livello di rumore.

L'input effettivo ricevuto dalla U-Net di denoising è costituito da:

- le immagini mascherate;
- la maschera binaria;
- le immagini con il rumore aggiunto nei pixel da ricostruire;
- *signal rate* e *noise rate*.

La U-Net predice a questo punto l'immagine pulita (il segnale utile), e a partire da essa viene calcolato il rumore presente nei pixel corrotti in base alla formula 4.1.

Infine, viene calcolata la funzione di perdita e i pesi della rete vengono aggiornati. In aggiunta, i pesi vengono sincronizzati anche con un modello **EMA** (*Exponential Moving Average*), che mantiene una media esponenziale dei pesi aggiornati ad ogni passo. Questo approccio consente di ottenere predizioni più stabili e robuste durante la fase di inferenza.

4.3.2.1 Funzione di Loss

La **funzione di perdita** (*loss*) serve a quantificare quanto le predizioni della rete si discostano dai valori reali (*ground truth*). Durante l'addestramento, l'obiettivo del modello è minimizzare questa funzione, in modo che le immagini predette siano il più possibile simili a quelle originali o desiderate. In altre parole, la *loss* guida l'**ottimizzazione dei pesi** della rete.

In questo lavoro, la funzione di loss che ha portato ai risultati migliori è la **L1** (equivalente al MAE, *Mean Absolute Error*), definita come:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \,,$$

dove N è il numero totale dei pixel, y_i rappresenta il valore reale del pixel i e $\hat{y}_i$ quello predetto.

Siccome la L1 misura la distanza assoluta tra i pixel previsti e quelli reali, il contributo di ciascun errore al calcolo della funzione di perdita cresce linearmente alla propria ampiezza. Ciò implica che gli errori di piccola e grande entità influenzano la perdita in modo proporzionale, evitando che pochi errori molto grandi dominino il valore complessivo (come invece accade più frequentemente nella loss L2, equivalente al MSE). Questa proprietà consente di ottenere ricostruzioni con bordi più nitidi e un maggior livello di dettaglio.

Tali caratteristiche rendono la loss L1 particolarmente adatta, nello specifico, al compito di inpainting.

4.3.3 Inferenza

Dopo l'addestramento, il modello può essere utilizzato per generare nuove immagini ricostruite tramite **inferenza**, processo noto anche come *sampling*.

All'inizio del processo, nei pixel da ricostruire viene aggiunto rumore gaussiano casuale, mentre nelle regioni già visibili il segnale originale viene leggermente attenuato tramite un fattore *min_signal_rate*. Questo accorgimento permette di armonizzare anche le parti osservabili con lo schema di diffusione, introducendo un piccolo adattamento senza però aggiungere rumore significativo. Tale scelta si discosta da quanto comunemente adottato in letteratura, dove l'area visibile viene solitamente mantenuta invariata; in questo caso, tale tecnica si è rivelata vantaggiosa, poiché consente di ridurre la discontinuità tra regioni osservate e ricostruite, favorendo una maggiore coerenza visiva nell'immagine finale.

Successivamente, la rete applica il **processo inverso** di diffusione in un numero prefissato di passi. Ad ogni iterazione, la U-Net di denoising riceve in input un'immagine rumorosa e produce una stima del segnale utile (l'immagine pulita). A partire da questa predizione, è possibile calcolare anche la componente di rumore residuo.

Avviene a questo punto la fase di ***noise reinjection***, che utilizza i coefficienti *signal rate* e *noise rate*, calcolati come descritto nel paragrafo precedente. In pratica, la parte di segnale predetta dalla rete viene scalata in base al *signal rate*, mentre la componente di rumore stimata viene reinserita in misura proporzionale al *noise rate*. Formalmente:

$$x_{t-1} = \text{signal\_rate}_{t-1} \cdot \hat{x}_0 \; + \; \text{noise\_rate}_{t-1} \cdot \hat{\epsilon}$$

dove:

- $\hat{x}_0$ è l'immagine predetta dalla rete;
- $\hat{\epsilon}$ è il rumore stimato;
- signal_rate_{t-1} e noise_rate_{t-1} sono i coefficienti calcolati per il passo successivo di diffusione.

L'idea alla base segue il principio dei **DDIM (*Denoising Diffusion Implicit Models*)**: invece di introdurre nuovo rumore gaussiano casuale ad ogni passo (come avviene nei DDPM), viene reinserito lo stesso rumore precedentemente rimosso, ma in forma attenuata. Questo approccio rende il processo deterministico, a parità di rumore iniziale.

Al termine delle iterazioni, l'output finale corrisponde alla predizione dell'immagine ricostruita.

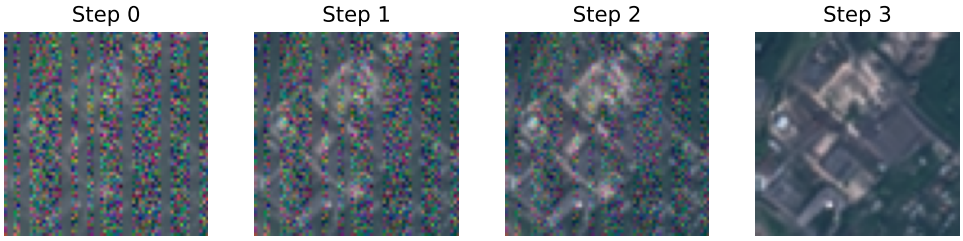

Figura 4.4: Visualizzazione di 4 step di diffusion durante l'inpainting di un'immagine del dataset EuroSAT/RGB.

4.4 Setup e procedura di training

Durante l'addestramento, il modello viene allenato per più **epoche**, ovvero per più passate complete sull'intero set di training. In ciascuna epoca, l'insieme di dati viene suddiviso in batch di dimensione predeterminata, e il modello aggiorna i propri pesi ad ogni batch in base alla funzione di loss calcolata. Questo processo permette alla rete di vedere ripetutamente tutti gli esempi di training, affinando progressivamente la capacità di generalizzazione e la qualità delle predizioni.

Gli iperparametri utilizzati sono riportati nella tabella 4.3. Come si può osservare, le dimensioni dei batch e il numero di epoche variano in base al dataset e al tipo di rete di denoising, mentre il learning rate iniziale è stato mantenuto costante a 1e−4 in tutti gli esperimenti.

Tabella 4.3: Iperparametri utilizzati durante il training

Iperparametro	**Inpainting (EuroSAT/RGB)**	**Colorizzazione (Oxford Flowers 102)**
Dimensione del batch	100	32
Learning rate iniziale	1e-4	1e-4
Numero di epoche (U-Net residuale)	200	125
Numero di epoche (Attention U-Net)	180	100

Nel training del presente modello è stato utilizzato l'ottimizzatore **Adam** (*Adaptive Moment Estimation*), capace di adattare automaticamente i tassi di apprendimento

dei singoli pesi. Per facilitare la convergenza, è stata inoltre impiegata una strategia di **Reduce LR on Plateau**, che riduce il learning rate quando la funzione di perdita calcolata non migliora per un numero prefissato di epoche, evitando stagnazioni dell'addestramento.

Durante il training, i pesi della rete sono stati salvati e ricaricati più volte, al fine di monitorare più accuratamente i progressi e acquisire un maggiore "feeling" sull'andamento della loss e della qualità delle immagini generate. In particolare, sono state monitorate sia la *image loss*, che misura l'accuratezza della ricostruzione dell'immagine finale, sia la *noise loss*, che valuta la precisione della predizione del rumore residuo.

Capitolo 5

Valutazione del modello di diffusione e confronto con U-Net

5.1 Metriche di valutazione

In questa sezione vengono introdotte le metriche utilizzate per la valutazone delle prestazioni del modello.

5.1.1 Mean Squared Error (MSE)

La metrica **MSE** misura la media dei quadrati delle differenze tra i valori previsti dal modello e quelli reali. In ambito di image restoration, questa metrica valuta quanto l'immagine ricostruita si discosta da quella originale pixel per pixel. Un valore di MSE più basso indica una maggiore accuratezza della ricostruzione. La formula è la seguente:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(I_i - K_i)^2$$

dove I_i è il valore del pixel originale, K_i è il valore del pixel ricostruito e N è il numero totale di pixel.

5.1.2 Peak Signal-to-Noise Ratio (PSNR)

La metrica **PSNR** è utilizzata per quantificare la qualità di un'immagine ricostruita rispetto all'immagine originale, esprimendola in decibel (dB). Valori di PSNR più elevati indicano una maggiore somiglianza tra l'immagine originale e quella ricostruita, corrispondente a una minore quantità di rumore, mentre valori più bassi

indicano una maggiore perdita di qualità. Formalmente:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{\text{MSE}} \right)$$

dove MAX_I rappresenta il valore massimo possibile di un pixel (ad esempio 255 per immagini a 8 bit).

5.1.3 Structural Similarity Index (SSIM)

SSIM è una metrica percettiva che valuta la somiglianza tra due immagini considerando luminosità, contrasto e struttura locale, piuttosto che solo la differenza pixel per pixel. Fornisce un valore compreso tra -1 e 1, dove 1 indica immagini identiche. Questa metrica è particolarmente utile in image restoration perché riflette meglio la percezione visiva della qualità rispetto a MSE e PSNR. La formula dello SSIM tra due finestre (piccole sotto-aree dell'immagine su cui calcolare localmente la somiglianza) x e y di dimensione $N \times N$ è la seguente:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

dove:

- μ_x, μ_y sono le medie dei pixel nelle finestre x e y,
- σ_x^2, σ_y^2 sono le varianze locali,
- σ_{xy} è la covarianza tra x e y,
- C_1, C_2 sono costanti per stabilizzare la divisione quando i denominatori sono vicini a zero.

È importante sottolineare che un SSIM elevato riflette una maggiore omogeneità strutturale, ma non implica necessariamente una maggiore nitidezza visiva.

5.2 Risultati

Le **prestazioni del modello di diffusione** sono state confrontate con due **U-Net tradizionali**, ovvero semplici reti convoluzionali encoder–decoder con *skip connection*, senza meccanismi di attenzione né blocchi residuali. Una di queste è stata addestrata per il compito di inpainting, mentre l'altra per la color reconstruction. Questo confronto è stato pensato per mettere in evidenza i benefici introdotti dall'approccio di diffusione e, in particolare, dall'utilizzo delle tecniche di *ensemble*.

L'architettura U-Net è infatti uno degli approcci più consolidati per compiti di inpainting e colorizzazione, e costituisce quindi un termine di paragone naturale per valutare i miglioramenti introdotti dal modello a diffusione.

Sono stati effettuati due tipi di valutazione sulle prestazioni del modello:

- **Predizione singola**: per ciascuna immagine del test set (7000 campioni per EuroSAT/RGB e 1020 per Oxford Flowers 102) è stata generata una singola predizione. Le metriche MSE, SSIM e PSNR sono state calcolate sull'intero insieme e riportate come media e deviazione standard. Questo tipo di valutazione è stato effettuato anche sulle U-Net tradizionali.
- **Predizione ensemble**: per ciascuna immagine del test set sono state generate 32 predizioni indipendenti. Queste sono state successivamente combinate tra loro (tramite media pixel–wise) per ottenere una predizione finale più stabile e meno rumorosa. Le metriche MSE, SSIM e PSNR sono state quindi calcolate come nel caso precedente, ma sulle immagini risultanti dall'ensemble. Questa valutazione permette di evidenziare come la variabilità intrinseca del processo di diffusione possa essere sfruttata a vantaggio delle prestazioni complessive.

Ovviamente, per le U-Net non è stata effettuata la valutazione in modalità ensemble, poiché non trattandosi di modelli generativi la loro predizione è completamente deterministica: a un determinato input corrisponde sempre la stessa predizione.

Come si può osservare nelle Tabelle 5.1 e 5.2, le prestazioni del modello a diffusione, per quanto riguarda i valori di MSE, risultano in generale leggermente inferiori rispetto a quelle delle U-Net tradizionali quando si considera la predizione singola. Tuttavia, l'impiego della strategia di ensemble consente un significativo miglioramento: MSE si riduce sensibilmente, mentre PSNR e SSIM aumentano, portando a ricostruzioni più fedeli rispetto a quelle ottenute dalle U-Net standard.

Tabella 5.1: Inpainting: confronto tra U-Net, Diffusione Singola ed Ensemble Diffusion (32 predizioni) sul dataset EuroSAT/RGB. Risultati ottenuti con 4 step di diffusione.

Modello	**MSE (mean ± std)**	**SSIM (mean ± std)**	**PSNR (mean ± std)**
U-Net classica	0.002755 ± 0.000332	0.7995 ± 0.1294	28.86 ± 5.53
Diffusion singola (residual U-Net)	0.002879 ± 0.000389	0.8147 ± 0.1400	30.82 ± 5.28
Ensemble diffusion (residual U-Net)	0.002353 ± 0.000333	0.8365 ± 0.1246	31.64 ± 5.08
Diffusion singola (attention U-Net)	0.002848 ± 0.000395	0.8171 ± 0.1402	30.92 ± 5.30
Ensemble diffusion (attention U-Net)	0.002326 ± 0.000335	0.8387 ± 0.1241	31.72 ± 5.10

Tabella 5.2: Color reconstruction: confronto tra U-Net, Diffusione Singola ed Ensemble Diffusion (32 predizioni) sul dataset Oxford Flowers 102. Risultati ottenuti con 4 step di diffusione.

Modello	**MSE (mean ± std)**	**SSIM (mean ± std)**	**PSNR (mean ± std)**
U-Net classica	0.006755 ± 0.001076	0.8717 ± 0.0551	23.10 ± 3.43
Diffusion singola (residual U-Net)	0.007281 ± 0.000991	0.8720 ± 0.0548	22.60 ± 3.33
Ensemble diffusion (residual U-Net)	0.006241 ± 0.000913	0.8883 ± 0.0510	23.42 ± 3.51
Diffusion singola (attention U-Net)	0.006640 ± 0.000760	0.8851 ± 0.0523	23.67 ± 3.43
Ensemble diffusion (attention U-Net)	0.005803 ± 0.000737	0.9003 ± 0.0477	24.66 ± 3.61

Da questi dati si può anche notare come, nel caso dell'**inpainting**, i valori di SSIM e PSNR ottenuti con la diffusione singola siano superiori a quelli della U-Net classica, segnalando immagini più coerenti dal punto di vista percettivo e con un livello di rumore inferiore rispetto al segnale utile. Come mostrato nella figura 5.1, questo si traduce in ricostruzioni visivamente più nitide e ricche di dettagli: nonostante la U-Net ottenga valori di MSE leggermente migliori, le immagini generate dal modello a diffusione risultano qualitativamente superiori agli occhi dell'osservatore. Questo fenomeno è particolarmente lampante nel compito di inpainting, dove la diffusione permette di preservare meglio i contorni e le strutture locali rispetto alle soluzioni più uniformi e leggermente sfocate restituite dalla U-Net.

Un'ulteriore osservazione che emerge dalle tabelle 5.1 e 5.2 riguarda l'impatto dell'introduzione dei meccanismi di **self-attention** nella rete di denoising. Nel compito di **colorizzazione**, la rete deve inferire i colori corretti a partire dal contesto globale dell'immagine; in questo scenario, i meccanismi di self-attention risultano particolarmente efficaci, poiché consentono di catturare relazioni a lungo raggio tra regioni non adiacenti ma semanticamente correlate, come diverse parti dello stesso fiore che condividono la medesima colorazione. Questo spiega perché l'Attention U-Net riesca a superare di molto la U-Net standard in termini di prestazioni (nella valutazione ensemble).

Nel compito di **inpainting**, invece, l'obiettivo è ricostruire regioni mancanti basandosi soprattutto sul contesto locale circostante ai pixel oscurati. In questo caso, le informazioni globali fornite dall'attenzione hanno un ruolo meno determinante, mentre diventa cruciale mantenere la coerenza strutturale e la continuità dei dettagli. La semplice convoluzione, unita ai benefici introdotti dal *residual learning*, risulta in questo caso sufficiente: attention U-Net e residual U-Net producono risultati simili.

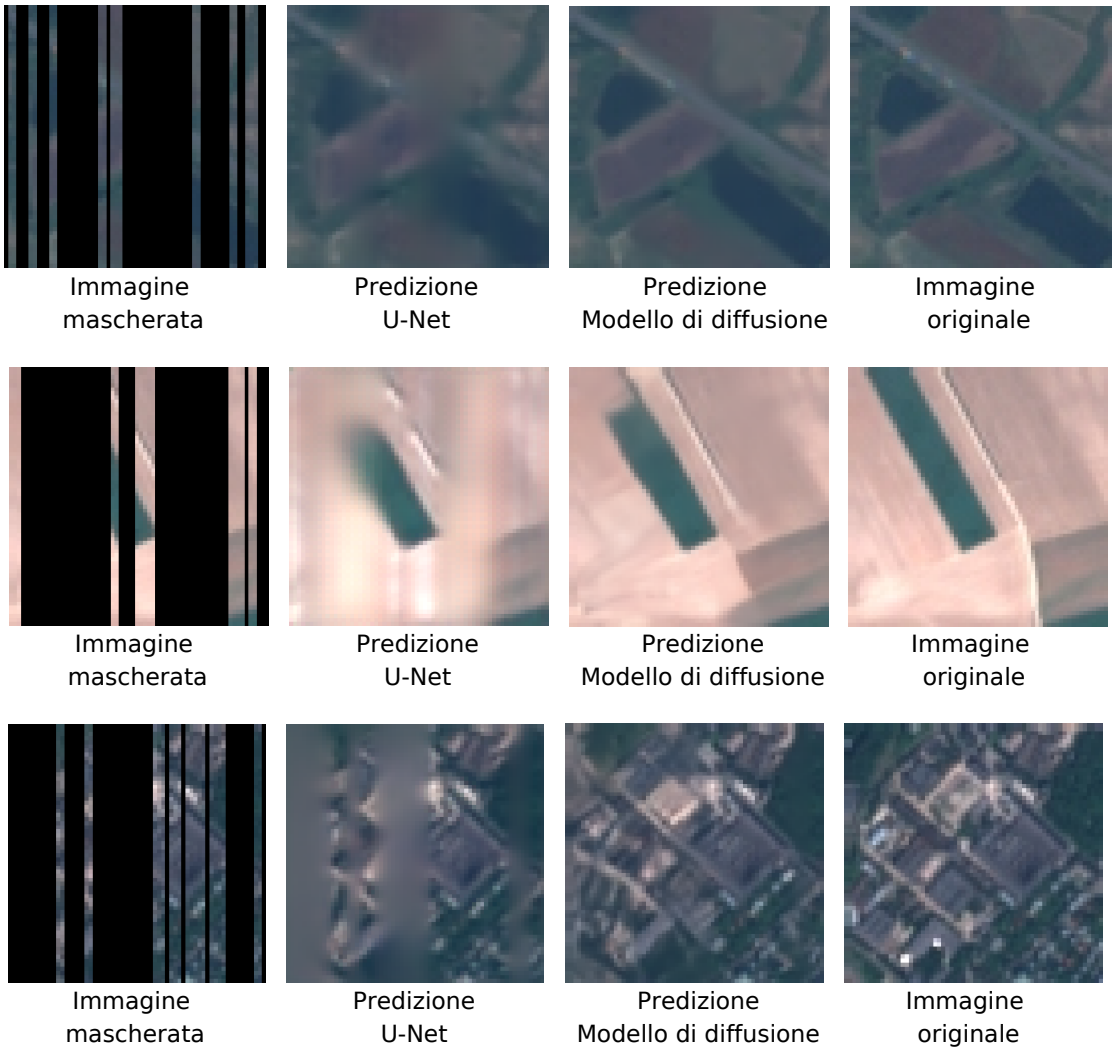

Figura 5.1: Confronto visivo tra le ricostruzioni ottenute dal modello di diffusione (predizione singola) e quelle prodotte dalla U-Net classica su tre esempi del test set.

Figura 5.2: Esempi di ricostruzioni dei canali RGB ottenute dal modello di diffusione (predizione singola) su tre immagini del test set.

5.2.1 Effetto del numero di predizioni nell'ensemble

Un primo esperimento ha analizzato come variano le metriche di valutazione al crescere del numero di predizioni mediate nell'ensemble.

Dai grafici riportati in appendice B si osserva che, all'aumentare del numero di predizioni, il valore di MSE diminuisce e il PSNR aumenta, segnalando una maggiore vicinanza numerica all'immagine di riferimento. Anche il valore di SSIM, pur essendo una metrica percettiva, mostra un incremento, poiché la coerenza strutturale e la distribuzione locale di luminosità e contrasto non vengono compromesse dall'ensemble, ma al contrario migliorano.

Nel caso dell'inpainting, tuttavia, la media delle predizioni comporta un effetto collaterale: l'attenuazione dei dettagli fini. Questo rende le immagini risultanti più

uniformi e leggermente sfocate, come visibile nell'immagine 5.3.

Figura 5.3: Risultati di inpainting su un'immagine del test set al variare del numero di predizioni mediate nell'ensemble: 1, 16 e 32.

.

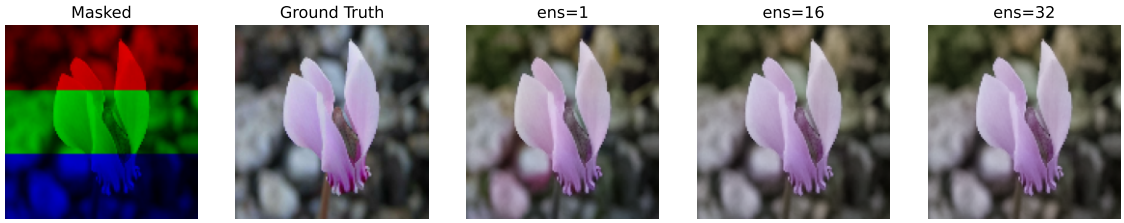

Figura 5.4: Risultati di color reconstruction su un'immagine del test set al variare del numero di predizioni mediate nell'ensemble: 1, 16 e 32.

.

5.2.2 Effetto del numero di passi di diffusione nell'inferenza

Un secondo esperimento ha valutato l'impatto del numero di *diffusion step* utilizzati durante l'inferenza.

Come mostrato nei grafici in appendice B, l'andamento delle metriche non risulta monotono ma presenta diverse oscillazioni. In generale, i risultati migliori si ottengono con un numero ridotto di passi (indicativamente meno di dieci). Una possibile spiegazione risiede nella *diffusion schedule* adottata in fase di training: invece di iterare sistematicamente tutti gli step della catena di diffusione, per ogni immagine viene simulato un livello di rumore scelto casualmente. Questo approccio sembra aver reso il modello particolarmente adatto a ricostruire immagini in pochi step di diffusione. Tale comportamento è in realtà coerente con quanto osservato in letteratura: come mostrato da [21], i *Denoising Diffusion Implicit Models* (DDIM) consentono di ridurre significativamente il numero di passi di inferenza rispetto ai tradizionali DDPM, mantenendo comunque una qualità di ricostruzione soddisfacente.

Capitolo 6

Conclusioni

L'obiettivo principale di questo lavoro era valutare i vantaggi della *diffusion inpainting* rispetto a tecniche più tradizionali, come l'inpainting eseguito con una semplice U-Net: dai risultati ottenuti emerge chiaramente come i modelli a diffusione siano in grado di generare immagini con una nitidezza e un livello di dettaglio superiore. Inoltre, l'adozione di tecniche di *ensemble* migliora significativamente le metriche quantitative (MSE, PSNR, SSIM), riducendo l'errore medio e producendo ricostruzioni più stabili.

La robustezza e la versatilità della tecnica sono state ulteriormente confermate applicando il modello a un diverso compito di image reconstruction: la colorizzazione dei canali RGB. In questo scenario, l'introduzione di meccanismi di *self-attention* nella rete di denoising si è rivelata particolarmente efficace, poiché ha permesso di catturare relazioni a lungo raggio tra regioni semanticamente correlate, migliorando ulteriormente la qualità delle ricostruzioni.

Un ulteriore vantaggio dei *Denoising Diffusion Implicit Models* (DDIM) rispetto ai classici DDPM risiede nella possibilità di generare nuove ricostruzioni velocemente e con un numero ridotto di passi di diffusione, mantenendo comunque alta la qualità delle immagini. Questo si traduce in un notevole risparmio computazionale durante la fase di inferenza.

Tuttavia, il lavoro presenta alcune limitazioni. In primo luogo, l'addestramento dei modelli a diffusione è computazionalmente più oneroso rispetto alle U-Net tradizionali, richiedendo tempi di training più lunghi e maggiore capacità di memoria. Lo stesso vale per le tecniche di *ensemble*: sebbene migliorino stabilità e metriche quantitative, esse comportano costi aggiuntivi significativi in fase di inferenza. Di conseguenza, il miglioramento ottenuto tramite ensemble va sempre bilanciato col vincolo pratico dei tempi di generazione e delle risorse disponibili.

Per futuri sviluppi, sono emersi diversi spunti interessanti. L'integrazione di architetture basate su *Transformer* al posto delle U-Net, sfruttando la capacità dei meccanismi di attenzione di catturare relazioni globali tra pixel, potrebbe migliorare ulteriormente le prestazioni e ampliare l'applicabilità del modello di diffusione a una gamma più ampia di compiti di image reconstruction, come la super-risoluzione o il deblurring. Altre possibili direzioni includono l'esplorazione di tecniche di *ensemble* più sofisticate.

In conclusione, i risultati ottenuti confermano che i modelli a diffusione rappresentano una valida alternativa alle tecniche classiche di inpainting, con vantaggi evidenti in termini qualitativi, e offrono numerose possibilità di estensione e miglioramento per lavori futuri.

Appendice A

Stato dell'arte del diffusion inpainting

In questa appendice sono riportate due tabelle di sintesi che raccolgono alcuni contributi rilevanti della letteratura recente sul *diffusion inpainting*, evidenziandone caratteristiche architetturali e strategie.

Tabella A.1: Caratteristiche dei modelli di diffusion inpainting in letteratura

Paper	**Modello**	**Diffusion step**	**Condizionamento**	**Loss**	**Metriche di valutazione**	**Dataset**
[9]	DDPM	2K (training), 1K (inferenza)	Maschera *free-form* + concatenazione	MSE, L1	IS, FID, CA, PD	Places2, ImageNet
[10]	DDPM	250	Maschera *free-form* (solo nell'inferenza)	MSE	LPIPS, *User study*	CelebA-HQ, ImageNet
[12]	DDPM	4	IPR	L1, Perceptual loss	FID, LPIPS	Places, CelebA-HQ
[14]	DDPM o DDIM preaddestrati	20	Matrice di degradazione lineare H, SVD $H = U\Sigma V^\top$	MSE, ELBO	PSNR, SSIM	ImageNet, CelebA-HQ, LSUN bedrooms, LSUN cats
[11]	DDPM (adattabile come DDIM)	2K	Perlin Noise Masks	MSE (variante *pixel-wise*)	FID, LPIPS	FFHQ, LSUN bedrooms, ImageNet
[17]	DDIM	250	Maschera *free-form* + approccio bayesiano	MSE	LPIPS, *User study*	CelebA-HQ, ImageNet
[13]	Stable Diffusion	20K	Maschera + testo + precisione della maschera	MSE	FID, *User study*	LAION-Aesthetics, OpenImages, MSCOCO
[22]	Basato su DDPM	1K	Maschera *free-form*	Non specificata	PSNR, SSIM, FID	CelebA-HQ, Places2

Tabella A.2: Strategie di inpainting in letteratura

Paper	Tecniche di inpainting	Innovazioni / peculiarità
[9]	Rumore Gaussiano solo nelle aree mascherate durante l'addestramento e l'inferenza.	-
[10]	Durante l'inferenza i pixel noti sono campionati direttamente dall'immagine in input, e vengono generati solamente quelli sconosciuti.	Modello pre-addestrato in modo incondizionato: gestisce qualsiasi forma di maschera e mantiene buone capacità semantiche; maschera utilizzata solo in fase di inferenza.
[12]	Nel training, non viene utilizzato un approccio tradizionale con maschera binaria; il modello sfrutta l'IPR (una rappresentazione degradata della ground truth) per guidare la generazione.	Tre componenti innovative: CPEN (estrae IPR dalle immagini ground-truth), DIRformer (U-Net con blocchi transformer dinamici che), Denoising Network (stima IPR da immagini degradate).
[14]	Inpainting sfruttando la SVD di H: componenti con valori singolari $s_i > 0$ vincolate ai pixel visibili, componenti con $s_i = 0$ ricostruite dal modello generativo pre-addestrato	Apprendimento non supervisionato; SVD-based diffusion; non serve training specifico per ogni nuova H
[11]	Rumore Gaussiano solo nelle aree mascherate durante l'addestramento e l'inferenza.	Embedding globale del rumore sostituito con mappe di rumore $H \times W$ per adattamento *pixel-wise* del denoising.
[17]	Utilizza framework bayesiano per ricostruire sia regioni visibili che nascoste.	Ricostruzione congiunta tra parte visibile e generata.
[13]	Rumore Gaussiano solo nelle aree mascherate durante l'addestramento e l'inferenza.	Combina inpainting e generazione testo-immagine.
[22]	Forward process degrada tutte le regioni non mascherate a rumore puro; reverse process usa MFFM e VSSM per guidare il restauro.	MFFM: estrae feature locali (dettagli) e globali per guidare la ricostruzione. VSSM: crea associazioni tra regioni mascherate e non mascherate, migliorando la coerenza semantica.

Appendice B

Analisi delle metriche al variare dei parametri

In questa appendice sono raccolti i grafici relativi all'andamento delle metriche di valutazione (MSE, SSIM e PSNR) al variare di due parametri chiave: il numero di predizioni nell'ensemble e il numero di diffusion step. L'obiettivo è evidenziare come la variazione di tali parametri influenzi le prestazioni del modello.

Si tenga presente che:

- i grafici relativi all'andamento delle metriche al variare del numero di step di diffusione sono stati creati considerando la predizione singola, e dunque non l'approccio ensemble;
- i grafici relativi all'andamento delle metriche al variare del numero di predizioni dell'ensemble sono stati ottenuti impiegando 4 step di diffusione nell'inferenza;
- tutti i grafici riportati sono stati ottenuti a partire dai pesi del modello addestrato con la U-Net residuale come rete di denoising.

B.1 Andamento delle metriche per l'inpainting

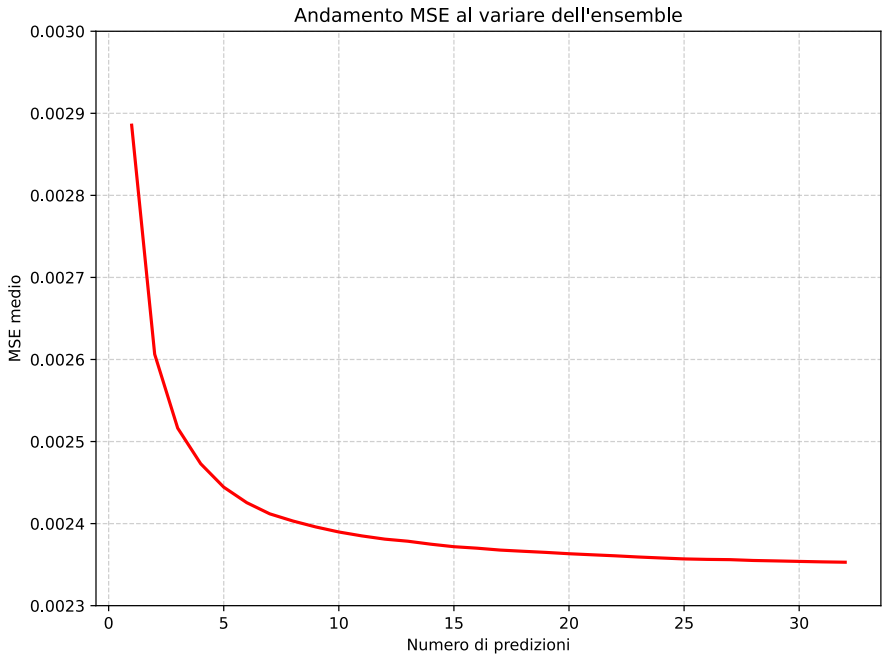

Figura B.1: Inpainting: andamento della metrica MSE al variare dell'ensemble

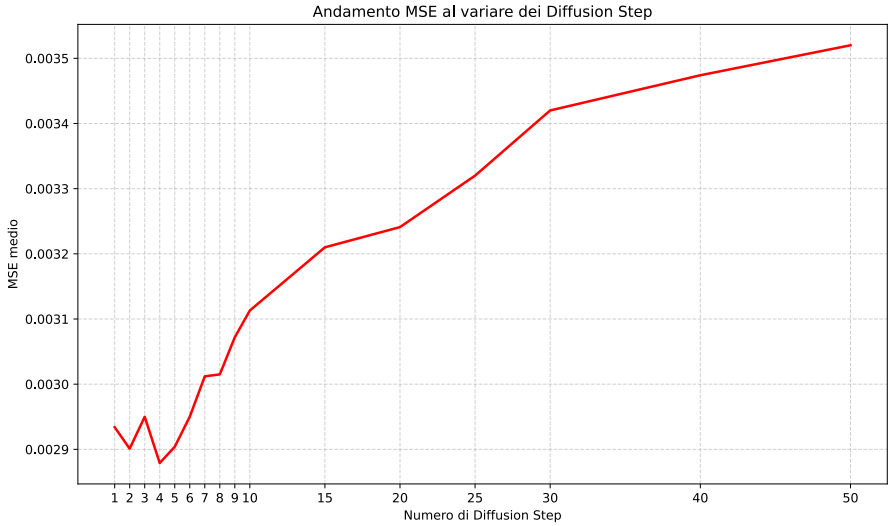

Figura B.2: Inpainting: andamento della metrica MSE al variare del numero di diffusion step

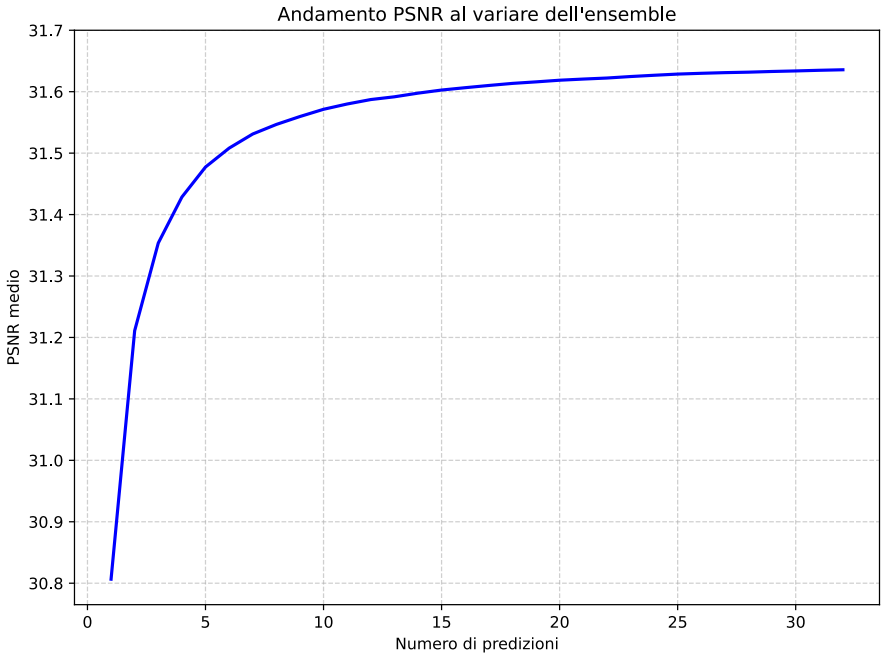

Figura B.3: Inpainting: andamento della metrica PSNR al variare dell'ensemble

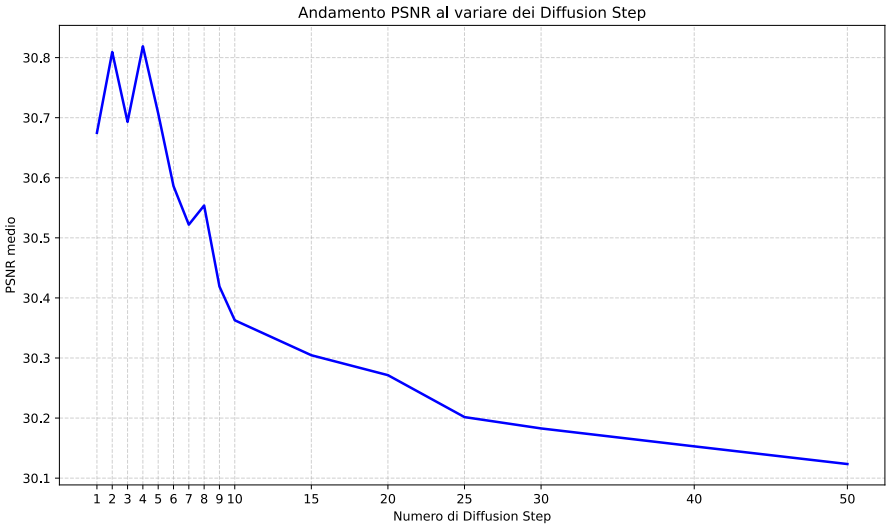

Figura B.4: Inpainting: andamento della metrica PSNR al variare del numero di diffusion step

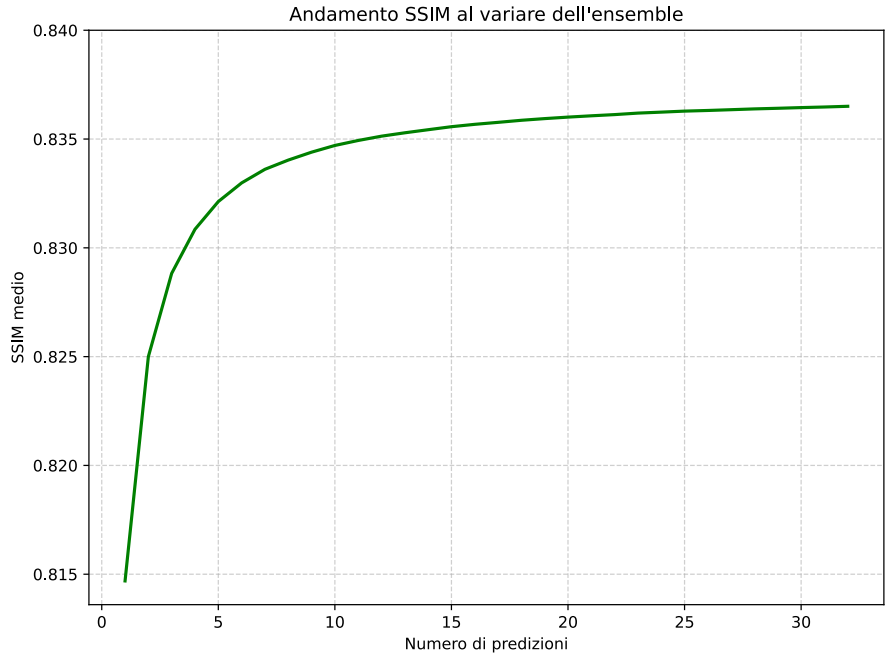

Figura B.5: Inpainting: andamento della metrica SSIM al variare dell'ensemble

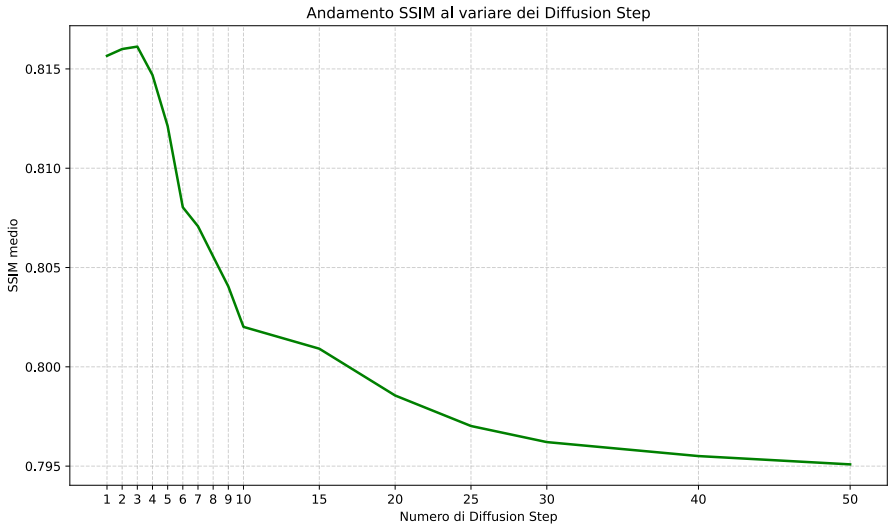

Figura B.6: Inpainting: andamento della metrica SSIM al variare del numero di diffusion step

B.2 Andamento delle metriche per la color reconstruction

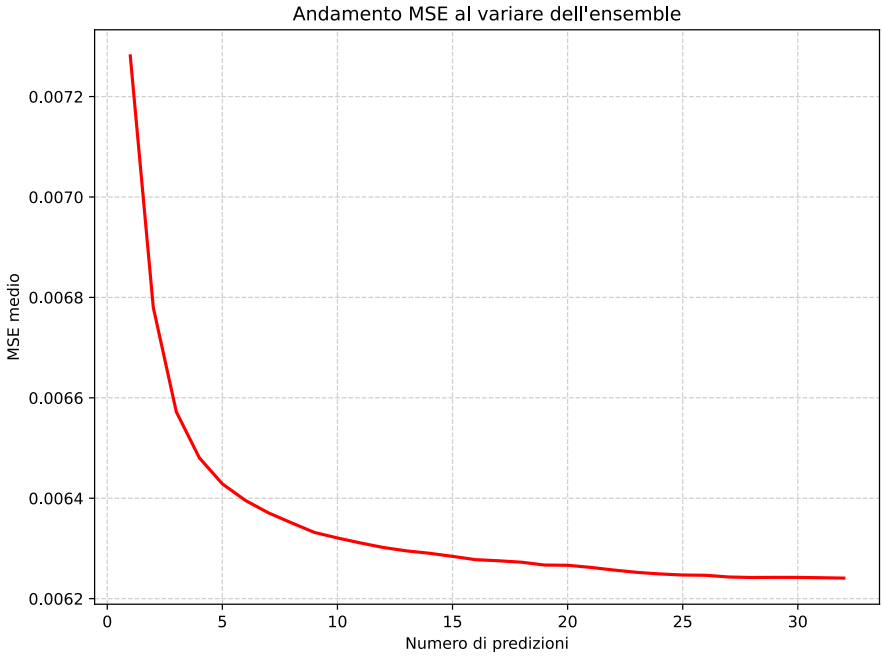

Figura B.7: Colorizzazione: andamento della metrica MSE al variare dell'ensemble

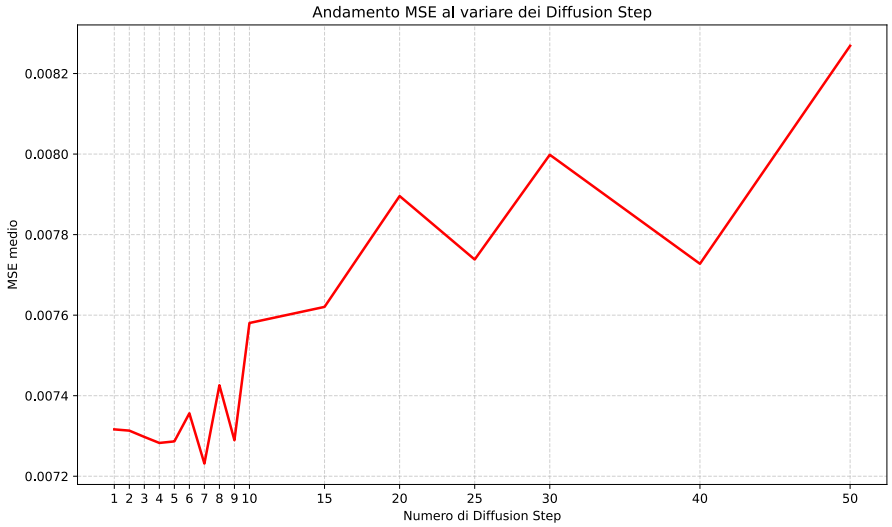

Figura B.8: Colorizzazione: andamento della metrica MSE al variare del numero di diffusion step

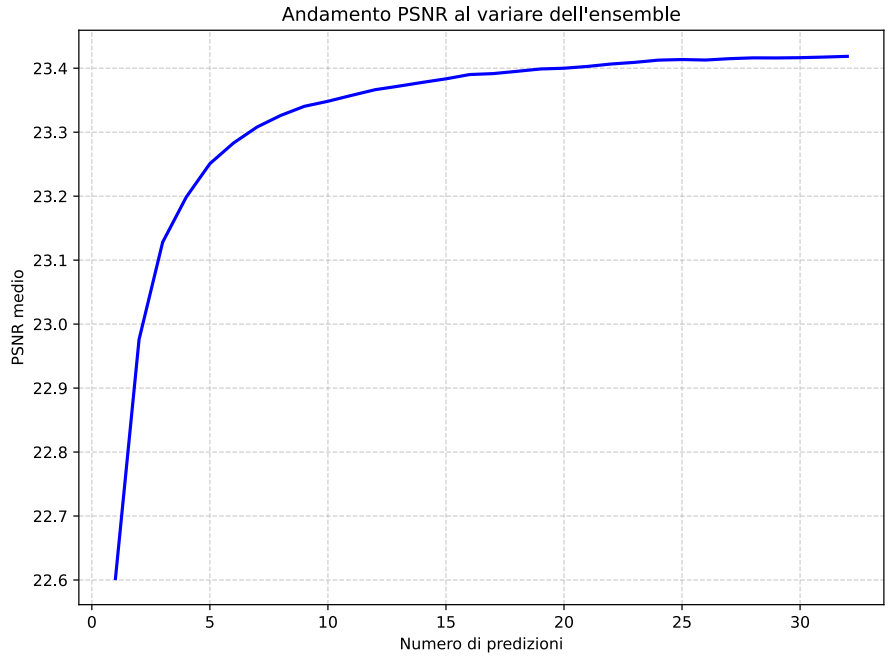

Figura B.9: Colorizzazione: andamento della metrica PSNR al variare dell'ensemble

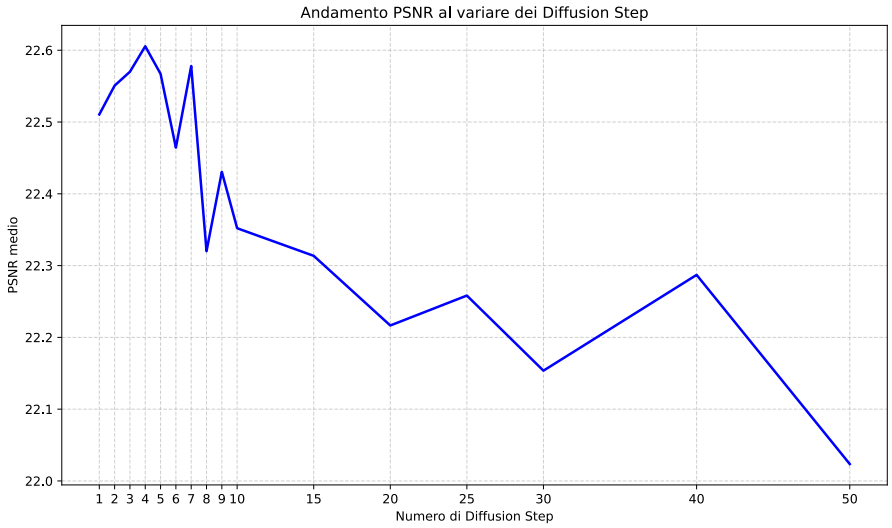

Figura B.10: Colorizzazione: andamento della metrica PSNR al variare del numero di diffusion step

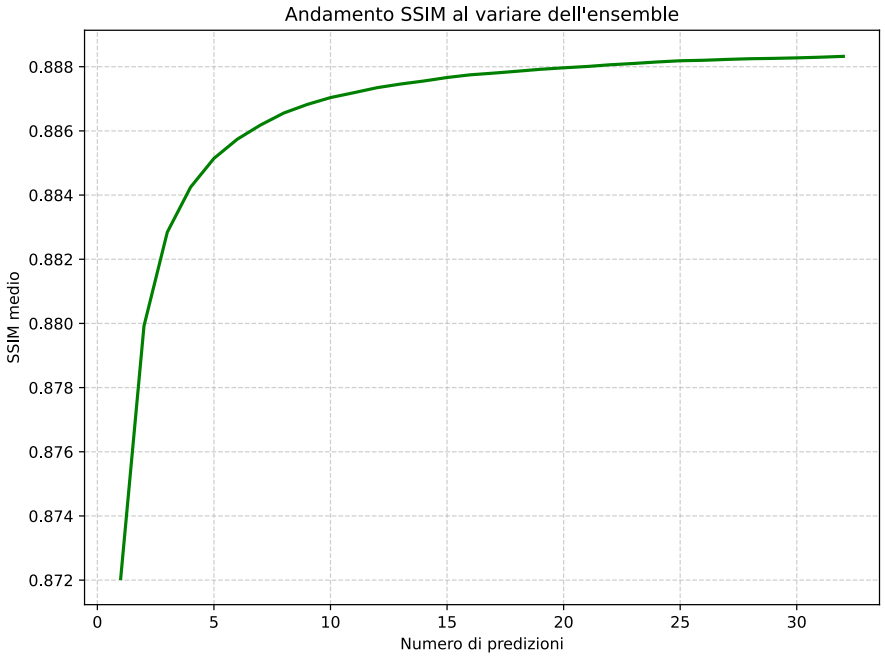

Figura B.11: Colorizzazione: andamento della metrica SSIM al variare dell'ensemble

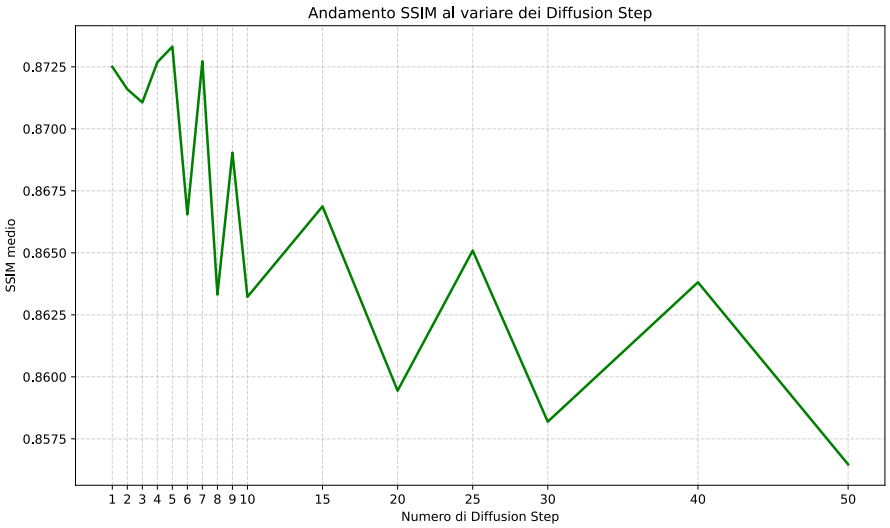

Figura B.12: Colorizzazione: andamento della metrica SSIM al variare del numero di diffusion step

Bibliografia

[1] Zihan Liu. Literature review on image restoration. In *Journal of Physics: Conference Series, Volume 2386, December 2022*, page 012041. IOP Publishing, 2022. https://iopscience.iop.org/article/10.1088/1742-6596/2386/1/012041.

[2] Balamurugan Palanisamy, Vikas Hassija, Arpita Chatterjee, Arpita Mandal, Debanshi Chakraborty, Amit Pandey, G. Sai Sesha Chalapathi, and Dhruv Kumar. Transformers for vision: A survey on innovative methods for computer vision. *IEEE Access*, 13:95496–95523, 2025. https://doi.org/10.1109/ACCESS.2025.3571735.

[3] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. https://doi.org/10.1007/978-3-319-24574-4_28.

[5] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2023. http://udlbook.com.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[7] Lilian Weng. What are diffusion models?, 2021. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[9] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In Munkhtsetseg Nandigjav, Niloy J. Mitra, and Aaron Hertzmann, editors, *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, pages 15:1–15:10. ACM, 2022. https://doi.org/10.1145/3528233.3530757.

[10] Andreas Lugmayr, Martin Danelljan, Andrés Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11451–11461. IEEE, 2022. https://doi.org/10.1109/CVPR52688.2022.01117.

[11] Sora Kim, Sungho Suh, and Minsik Lee. RAD: region-aware diffusion models for image inpainting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 2439–2448. Computer Vision Foundation / IEEE, 2025. https://openaccess.thecvf.com/content/CVPR2025/html/Kim_RAD_Region-Aware_Diffusion_Models_for_Image_Inpainting_CVPR_2025_paper.html.

[12] Bin Xia, Yulun Zhang, Shiyin Wang, Yitong Wang, Xinglong Wu, Yapeng Tian, Wenming Yang, and Luc Van Gool. Diffir: Efficient diffusion model for image restoration. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 13049–13059. IEEE, 2023. https://doi.org/10.1109/ICCV51070.2023.01204.

[13] Shaoan Xie, Zhifei Zhang, Zhe Lin, Tobias Hinz, and Kun Zhang. Smartbrush: Text and shape guided object inpainting with diffusion model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 22428–22437. IEEE, 2023. https://doi.org/10.1109/CVPR52729.2023.02148.

[14] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. http://papers.nips.cc/paper_files/paper/2022/hash/95504595b6169131b6ed6cd72eb05616-Abstract-Conference.html.

[15] Zhenning Shi, Haoshuai Zheng, Chen Xu, Changsheng Dong, Bin Pan, Xueshuo Xie, Along He, Tao Li, and Huazhu Fu. Resfusion: Denoising diffusion probabilistic models for image restoration based on prior residual noise. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. http://papers.nips.cc/paper_files/paper/2024/hash/ebc62a3af9342eb4ebc728e5c5bc4cca-Abstract-Conference.html.

[16] Tomer Garber and Tom Tirer. Image restoration by denoising diffusion models with iteratively preconditioned guidance. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 25245–25254. IEEE, 2024. https://doi.org/10.1109/CVPR52733.2024.02385.

[17] Guanhua Zhang, Jiabao Ji, Yang Zhang, Mo Yu, Tommi S. Jaakkola, and Shiyu Chang. Towards coherent image inpainting using denoising diffusion implicit models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 41164–41193. PMLR, 2023. https://proceedings.mlr.press/v202/zhang23q.html.

[18] Hamadi Chihaoui and Paolo Favaro. Invert2restore: Zero-shot degradation-blind image restoration. *CoRR*, abs/2503.21486, 2025. https://doi.org/10.48550/arXiv.2503.21486.

[19] Matteo Turisini, Giorgio Amati, and Mirko Cestari. LEONARDO: A pan-european pre-exascale supercomputer for HPC and AI applications. *CoRR*, abs/2307.16885, 2023. https://doi.org/10.48550/arXiv.2307.16885.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep re-

sidual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. https://doi.org/10.1109/CVPR.2016.90.

[21] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. https://openreview.net/forum?id=St1giarCHLP.

[22] Linxu Chen, Zhiqing Guo, Liejun Wang, and Ke Lu. Blurpaint: Image inpainting using blurring diffusion models. In *2025 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2025, Hyderabad, India, April 6-11, 2025*, pages 1–5. IEEE, 2025. https://doi.org/10.1109/ICASSP49660.2025.10889066.

Ringraziamenti

È doveroso spendere due parole per esprimere la mia gratitudine a chi mi ha sostenuta, incoraggiata e accompagnata durante questo percorso.

Innanzitutto, un sentito ringraziamento al professor Asperti, il cui supporto e i cui preziosi consigli hanno reso possibile la realizzazione di questa tesi.

Ai miei compagni di corso, in particolare Chiara, Li, Cino, Bruni, presenti sin dai tempi delle superiori: senza di voi e la leggerezza che portate, questo percorso sarebbe stato certamente più difficile.

A Simone, per aver condiviso con me gioie e ansie di questi anni, per le ore di studio insieme e per la pazienza e il sostegno costante.

Ai miei genitori, per avermi sempre supportata (e sopportata).

Ai miei gattini Ade e Kora, per la molto apprezzata compagnia; alle mie cocorite Gin e Sherry, per il sottofondo musicale (non sempre richiesto); a Osvaldo, per avermi osservata tutto il tempo dall'acquario.

A tutti voi: grazie.