ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

Robustezza dei sistemi di watermarking per immagini generate

Tesi di laurea in:
Visione Artificiale

Relatore
Prof. Matteo Ferrara

Candidato
Giuseppe Fusco

Correlatori
Dott. Serafino Pandolfini
Dott. Lorenzo Pellegrini

I Sessione di Laurea
Anno Accademico 2025-2026

Ai miei supereroi senza mantello,
ai miei colossi in un mare in tempesta,
ai miei genitori.

Indice

Introduzione

Negli ultimi anni, la rapida diffusione di sistemi generativi basati su reti neurali profonde ha reso sempre più accessibile la produzione automatica di contenuti visivi di elevata qualità, aprendo nuove opportunità creative ma sollevando al contempo interrogativi urgenti sull'autenticità e sulla tracciabilità dei contenuti digitali. In risposta, i principali attori del settore hanno sviluppato sistemi di watermarking progettati per incorporare una firma invisibile direttamente nel processo generativo, così da consentire la verifica dell'origine anche dopo la distribuzione dell'immagine.

La presente tesi analizza la robustezza di due dei principali sistemi di watermarking per immagini generate da intelligenza artificiale, *SynthID* di Google DeepMind e *PixelSeal* di Meta, nei confronti di tecniche di rimozione sia tradizionali sia basate su apprendimento automatico, valutando fino a che punto queste firme digitali possano essere degradate mantenendo una qualità visiva accettabile dell'immagine. Il contributo principale consiste in una valutazione sperimentale comparativa dei due sistemi, condotta mediante protocolli di attacco distinti e metriche omogenee, al fine di evidenziarne i punti di forza e i limiti rispettivi.

Il lavoro è strutturato come segue. Il Capitolo 1 introduce il contesto tecnologico, descrivendo i principali modelli generativi per immagini e lo stato dell'arte nel watermarking nell'era della generative AI, con particolare attenzione agli standard emergenti. Il Capitolo 2 presenta la metodologia sperimentale: il dataset realizzato per questo studio, l'analisi tecnica dei due sistemi e le tecniche di attacco considerate. Per SynthID viene descritta la pipeline di rimozione spettrale reverse-SynthID; per PixelSeal vengono presentati separatamente l'attacco mediante rete U-Net di denoising, nelle sue due configurazioni, e una batteria indipendente di quarantotto attacchi tradizionali. Il Capitolo 3 riporta i risultati sperimentali e

ne discute le implicazioni, confrontando quantitativamente l'efficacia dei diversi approcci di rimozione e analizzando la robustezza dei due sistemi rispetto agli attacchi considerati.

Capitolo 1

Stato dell'Arte

1.1 Modelli Generativi per Immagini

Nel panorama dell'intelligenza artificiale, i modelli generativi rappresentano una classe di algoritmi progettati per la produzione autonoma di nuovi contenuti. L'obiettivo fondamentale di questi sistemi consiste nell'apprendere in maniera approfondita la distribuzione di probabilità sottostante a un insieme di dati di addestramento. L'efficacia di questi modelli si estende a diversi settori tecnologici, spaziando dallo sviluppo di soluzioni audio e video fino alle pipeline di simulazione e sintesi dei dati.

1.1.1 Autoencoder

Un autoencoder [1] è un'architettura neurale non supervisionata progettata per apprendere una rappresentazione compressa e a bassa dimensionalità, definita spazio latente o *bottleneck*, dei dati in ingresso. Il processo di addestramento si basa su un meccanismo di ottimizzazione vincolato, in cui i parametri della rete, ovvero pesi e bias di encoder e decoder, vengono estratti minimizzando una *loss function* che quantifica lo scostamento tra l'input originale e la sua ricostruzione.

Un autoencoder è composto da due reti più piccole, come rappresentato in figura 1.1: un **encoder** e un **decoder**. Durante l'addestramento, l'encoder apprende un insieme di feature, detta "rappresentazione latente", dai dati di input. Nel contempo, il decoder viene addestrato per ricostruire i dati sulla base di queste

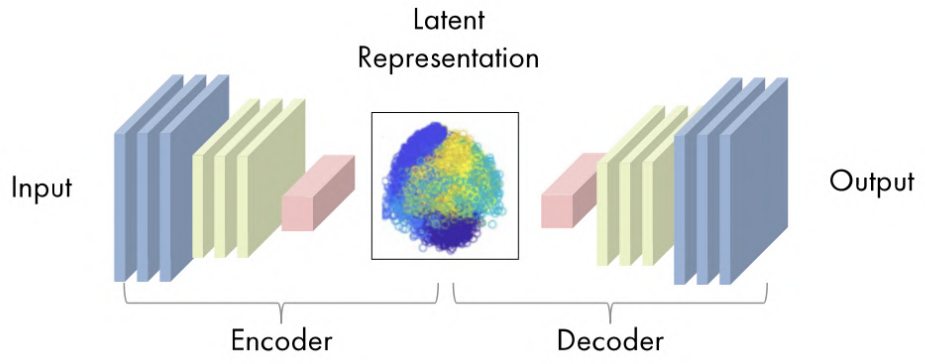


Figura 1.1: Struttura di un autoencoder

feature. In seguito, l'autoencoder può essere applicato per ricostruire input mai visti in precedenza.

Tra le diverse architetture si distinguono i **Variational Autoencoder (VAE)** [2], modelli generativi probabilistici in grado di sintetizzare nuovi dati coerenti con la distribuzione di addestramento. Dal punto di vista teorico, un VAE integra l'apprendimento profondo con l'inferenza bayesiana variazionale [3]: l'intera struttura mappa un unico framework probabilistico in cui un *encoder* stima i parametri della distribuzione variazionale condizionata, definendo uno spazio latente continuo e stocastico. Da quest'ultimo vengono campionati i vettori che il *decoder* utilizzerà per approssimare la distribuzione dei dati originali.

Pertanto, l'*encoder* mappa l'immagine in una distribuzione di probabilità all'interno dello spazio latente, anziché proiettarla in un singolo punto deterministico. D'altra parte, il *decoder* campiona i vettori da questa regione probabilistica per ricostruire l'asset originale (Figura 1.2).

Questo meccanismo introduce un vincolo di regolarizzazione intrinseco fondamentale per le capacità di generalizzazione del modello [3]. Negli autoencoder tradizionali, la rete tende a "memorizzare" i dati di addestramento associando a ogni immagine un punto isolato nello spazio; questo crea uno spazio latente frammentato, caratterizzato da ampie zone vuote che, se campionate, generano output geometricamente distorti o privi di significato semantico, anche detto fenomeno di *overfitting* geometrico.

Nei VAE, l'introduzione di una forza di regolarizzazione, dettata dal termine

della divergenza di Kullback-Leibler nella funzione obiettivo [3], costringe le distribuzioni prodotte dall'encoder a sovrapporsi e a strutturarsi attorno al centro dello spazio latente, conformandosi a una forma nota e regolare come una Gaussiana standard. Questo processo impedisce la formazione di cluster disgiunti o isolati, garantendo due proprietà topologiche cruciali: la **continuità**, ovvero punti vicini nello spazio latente vengono mappati su immagini con caratteristiche visive simili, e la **completezza**, ovvero qualsiasi area dello spazio latente racchiude informazioni coerenti e viene correttamente decodificata in un output sensato. L'intera architettura viene infine addestrata in modo congiunto sfruttando il *reparameterization trick* [3] per consentire il corretto passaggio dei gradienti attraverso i nodi stocastici.

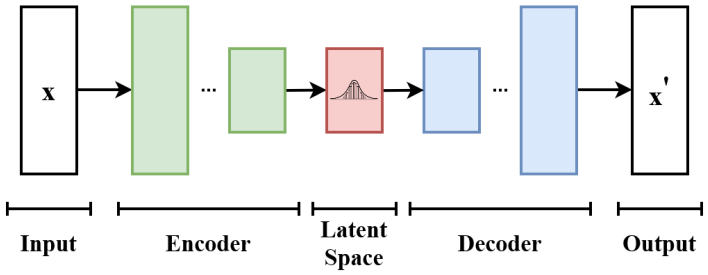


Figura 1.2: Architettura di un VAE: Il modello riceve x come input. L'encoder lo comprime nello spazio latente. Il decodificatore riceve come input le informazioni campionate dallo spazio latente e produce x' il più simile possibile a x.

1.1.2 Generative Adversarial Network

Una Generative Adversarial Network (GAN) è una classe di modelli generativi di machine learning. Le GAN sono state introdotte inizialmente da Ian Goodfellow e dai suoi colleghi nel giugno 2014 [4].

Dato un training set, questa tecnica impara a generare nuovi dati con le stesse statistiche dell'insieme di addestramento. Per esempio, una GAN addestrata su fotografie può generare nuove fotografie che appaiono, almeno superficialmente, autentiche a osservatori umani, possedendo molte caratteristiche realistiche. Sebbene originariamente proposte come una forma di modello generativo per l'apprendimento non supervisionato, le GAN sono state estese a contesti di semi-

supervised learning, supervised learning, solo per alcune varianti di GAN, e per il reinforcement learning.

L'idea centrale di una GAN si basa sull'interazione di due componenti principali (Figura 1.3) [5]:

- Il **generatore** impara a produrre dati plausibili, le cui istanze generate fungono da esempi di addestramento negativi per il discriminatore.
- Il **discriminatore** impara a distinguere i dati falsi provenienti dal generatore dai dati reali. Dal punto di vista computazionale, il discriminatore funge da loss function per il generatore: l'errore di classificazione commesso sulle immagini sintetiche genera un gradiente che, propagandosi a ritroso (*backpropagation*) attraverso la rete, permette di aggiornare i pesi del generatore, guidandolo verso la sintesi di contenuti statisticamente indistinguibili da quelli reali.

All'inizio della fase di addestramento, il generatore produce dati non strutturati che il discriminatore impara rapidamente a identificare come sintetici. Man mano che il processo avanza, il generatore ottimizza i propri pesi, guidato dai gradienti retropropagati, fino a produrre output in grado di simulare accuratamente le caratteristiche del dataset originale. Se il processo di ottimizzazione congiunta procede correttamente, il sistema converge verso un punto di equilibrio stocastico [5].

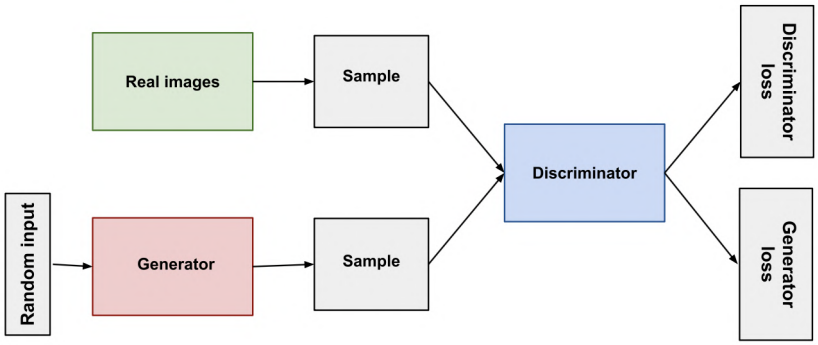


Figura 1.3: Funzionamento di una GAN

1.1.3 Modelli di Diffusione (Diffusion Models)

Nel campo del machine learning, i diffusion model, introdotti da Sohl-Dickstein e colleghi nel 2015 [6], rappresentano una classe di modelli generativi. Si compone di due elementi principali: il **forward/diffusion process** e il **reverse process**. L'obiettivo di tali architetture risiede nell'apprendere un processo di diffusione per un determinato dataset, in modo tale che il sistema sia in grado di generare nuovi elementi distribuiti in modo analogo al dataset originale. Da un modello di diffusione addestrato è possibile campionare in molteplici modalità, caratterizzate da differenti livelli di efficienza e qualità. Il modello responsabile della denoising è tipicamente chiamato "backbone", che in genere sono U-net o transformers.

I diffusion model vengono impiegati principalmente in compiti di computer vision, che includono il denoising di immagini, l'inpainting, la super-resolution e la generazione di immagini e video. Tali operazioni prevedono solitamente l'addestramento di una rete neurale finalizzata a rimuovere sequenzialmente il rumore da immagini tramite l'applicazione di rumore gaussiano [7]. La rete viene specificamente ottimizzata per invertire il processo di aggiunta del rumore a un'immagine. Una volta completato l'addestramento fino a convergenza, il modello può essere utilizzato per la generazione di contenuti partendo da un'immagine composta esclusivamente da rumore casuale e applicando la rete in modo iterativo per ripulire l'asset visivo. Il funzionamento è illustrato in Figura 1.4

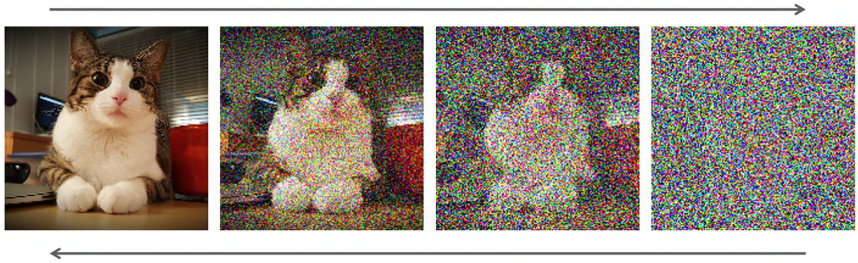


Figura 1.4: Meccanismo dei Modelli di Diffusione: il processo in avanti (*forward*) corrompe il dato, mentre il processo inverso (*reverse*) opera il *denoising*.

1.2 Watermarking nell'Era della Generative AI

La proliferazione di modelli generativi capaci di sintetizzare immagini iperrealistiche ha radicalmente trasformato il concetto di sicurezza multimediale, introducendo sfide inedite legate all'autenticità dei contenuti, alla proliferazione di **deepfake** e alla violazione del diritto d'autore. In questo scenario, il **watermarking**, formalmente definito come il processo di codifica e inserimento di un segnale informativo all'interno di un file multimediale o di altro genere, tale da poter essere successivamente rilevato o estratto per trarre informazioni sulla sua origine e provenienza, ha subito una profonda evoluzione. Nelle sue forme tradizionali, esso si manifestava come una sovrapposizione visibile, un logo, una filigrana o un testo, oppure come l'inserimento di metadati testuali nel file. Il watermarking moderno si discosta radicalmente da questo approccio: il segnale viene codificato in maniera impercettibile direttamente nel contenuto, risultando robusto alle comuni distorsioni e strettamente integrato con le pipeline di elaborazione neurale.

Un problema centrale è distinguere se un'immagine sia reale o generata da modelli sintetici. Nel contesto digitale classico, il watermarking è stato a lungo concepito come un'operazione di iniezione ex-post su un media già esistente [8]. Questo approccio tradizionale si muoveva principalmente su due binari: l'ap-

plicazione di filigrane visibili superficiali, facilmente rimovibili tramite ritaglio o mascheramento, o l'inserimento di stringhe informative all'interno dei metadati del file. Anche nelle sue varianti invisibili più avanzate, come la manipolazione dei bit meno significativi (Least Significant Bit, LSB) o la modifica dei coefficienti spettrali tramite trasformate discrete, il watermark tradizionale agiva come un elemento inserito in un secondo momento su una matrice di pixel rigida e già consolidata [8].

Al contrario, un sistema di watermarking moderno progettato per l'era della *Generative AI* non si configura come un'estensione successiva, ma si integra nativamente nel processo di sintesi. La differenza fondamentale risiede nel superamento della marcatura ex-post nello spazio dei pixel, intrinsecamente vulnerabile ed esposta a degradazione estetica.

Il nuovo metodo persegue il radicamento (*rooting*) del segnale direttamente nei pesi del modello generativo, ottimizzando preventivamente il decoder del VAE. L'immagine viene così sintetizzata inglobando il watermark all'interno delle sue stesse *feature* semantiche, rendendo la firma indissolubile dalla struttura matematica del generatore [9].

I vincoli tecnici per un sistema di watermarking moderno sono estremamente stringenti. Sebbene la caratterizzazione generale di un sistema di watermarking poggi su uno spettro più ampio di proprietà operative [8], lo scenario applicativo della *Generative AI* restringe il campo d'azione a un trade-off fondamentale governato principalmente da tre variabili critiche:

1. **Invisibilità (Imperceptibility):** La perturbazione introdotta non deve alterare la qualità percettiva o l'estetica dell'immagine, mantenendo inalterata l'esperienza dell'utente finale.

2. **Robustezza (Robustness):** Il watermark deve sopravvivere alle comuni operazioni di post-processing (compressione JPEG, resizing, cropping, noise filtering) e a tentativi di attacco malevolo volti alla sua rimozione.

3. **Capacità (Capacity):** La quantità di informazione codificabile nel watermark deve essere abbastanza grande da identificare univocamente il fornitore del servizio o, in alcuni scenari, l'identificativo della generazione.

Per soddisfare questi requisiti, la letteratura scientifica e i principali attori industriali del settore, quali possono essere Google DeepMind, Meta e OpenAI, si sono divise su due approcci metodologici principali, distinti dal momento esatto in cui il watermark viene impresso nel ciclo di vita dell'immagine: le tecniche *Post-Generation* (o *post-hoc*), che intervengono come layer successivo sulla mappa di pixel finita [9, 10], e le tecniche *In-Generation*, che integrano il watermark direttamente all'interno dei meccanismi matematici e degli spazi latenti del processo generativo [11, 9].

1.2.1 Watermarking Post-Generation

Il paradigma del watermarking post-generation prevede l'applicazione del segnale identificativo come un livello computazionale successivo ed esterno rispetto al ciclo di sintesi dell'immagine. In questo scenario, il modello generativo produce l'output multimediale; successivamente, una rete neurale secondaria interviene sul file generato per iniettare il payload informativo invisibile prima della sua distribuzione.

Dal punto di vista architetturale, le tecniche moderne basate su Deep Learning si affidano a una struttura ad autoencoder operante secondo uno schema di addestramento congiunto, di cui il framework *HiDDeN* (Figura 1.5) rappresenta uno dei principali capisaldi storici del deep-learning watermarking [10].

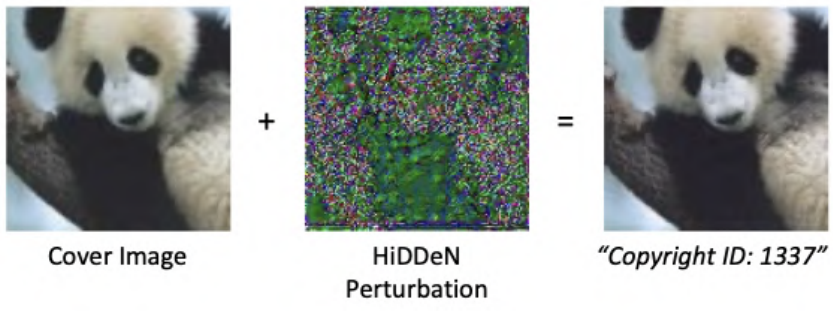


Figura 1.5: Dati un'immagine di copertina (cover image) e un messaggio binario, l'encoder di *HiDDeN* produce un'immagine codificata visivamente indistinguibile che contiene il messaggio, il quale può essere recuperato con un'elevata accuratezza da parte del decoder.

Il framework si configura come uno dei primi sistemi addestrabili end-to-end basato su tre reti neurali convoluzionali distinte e da un modulo di simulazione del rumore. I moduli fondamentali sono così strutturati:

- **Encoder:** Riceve in input la cover image I e un messaggio binario M, espresso come una stringa di bit. Genera in output l'immagine marcata $I_w = \text{Encoder}(I, M)$, minimizzando la distorsione visiva per rendere il segnale indistinguibile dall'originale.

- **Adversary (Rete Avversaria):** Analizza l'immagine e tenta di predire se essa contenga o meno un messaggio nascosto. Questo modulo genera una *adversarial loss* che migliora la qualità delle immagini prodotte dall'Encoder.

- **Layer di Distorsione (Noise Layer):** Interposto tra la fase di codifica e quella di decodifica, simula le alterazioni e le trasmissioni rumorose tipiche degli scenari reali (come la compressione lossy o trasformazioni geometriche). Genera una versione degradata I'_w per forzare il sistema ad apprendere codifiche robuste.

- **Decoder:** Riceve l'immagine alterata I'_w dal canale di trasmissione e tenta di ricostruire il payload originario $\hat{M}$. L'ottimizzazione mira a massimizzare l'accuratezza del recupero anche in presenza di forti degradazioni del segnale.

L'obiettivo globale di ottimizzazione del modello viene perseguito minimizzando congiuntamente: (1) la distanza tra l'immagine originale I e quella marcata I_w, (2) l'errore di ricostruzione tra il messaggio iniziale M e quello stimato $\hat{M}$, e (3) la capacità della rete avversaria di identificare la presenza del watermark.

Tuttavia, l'intrinseca separazione tra la fase di sintesi e quella di marcatura costituisce la sua principale vulnerabilità. Sebbene un watermark superficiale applicato sui pixel sia nativamente esposto ad attacchi di rimozione, l'introduzione di *noise layers* differenziabili, come *JPEG-Mask* e *JPEG-Drop* [10], in fase di addestramento forza l'encoder ad occultare il segnale in bande frequenziali geometricamente robuste, garantendo resilienza a compressioni lossy o ritagli estremi.

1.2.2 Watermarking In-Generation

A differenza delle tecniche di Post-Generation, il watermarking In-Generation (o *ingrained watermarking*) integra il segnale identificativo direttamente all'interno delle fasi di campionamento e dei processi stocastici del modello durante la sintesi stessa del contenuto [11]. Questa filosofia mira a rendere il watermark una caratteristica strutturale e statistica intrinseca alla distribuzione del dato generato, rendendo la sua separazione o rimozione estremamente complessa senza compromettere l'integrità qualitativa e la coerenza semantica dell'immagine (Figura 1.6).

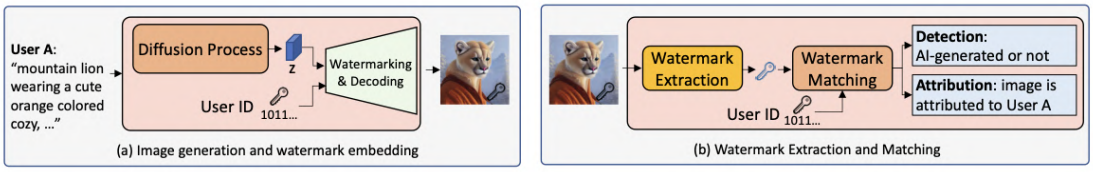


Figura 1.6: Panoramica dell'image watermarking in-generation per i Latent Diffusion Model (Latent Diffusion Model (LDM)). (a) Generazione dell'immagine e incorporamento del watermark. (b) Estrazione del watermark e confronto per l'identificazione e l'attribuzione dell'origine del contenuto.

Nei LDM, questo paradigma si implementa principalmente intervenendo su due componenti critiche della pipeline computazionale. Il primo approccio agisce sulle fasi di *encoding* e *decoding*, costringendo il modulo di decodifica a imprimere il segnale informativo durante la conversione dei dati dallo spazio latente a quello dei pixel. Il secondo, invece, agisce sullo scheduler e sulla strutturazione del rumore iniziale durante il processo di campionamento, imprimendo un pattern identificativo direttamente nel vettore latente di partenza in modo tale che l'immagine risultante incorpori il watermark in maniera rilevabile solo da un decoder dedicato.

Il punto di forza risiede nell'elevata robustezza: il watermark non è un'aggiunta superficiale, ma è incorporata all'interno delle immagini prodotte. Di contro, l'approccio in-generation presenta una complessità implementativa notevole, in quanto richiede il controllo diretto sulla pipeline di generazione, un potenziale overhead computazionale in fase di inferenza e una stretta dipendenza dall'architettura specifica del modello generativo sottostante.

1.3 Standard e Tecnologie Correnti

Di seguito viene condotta un'analisi tecnica approfondita sulle implementazioni commerciali e open-source adottate dai quattro principali attori tecnologici del settore.

1.3.1 Google DeepMind: SynthID

Sviluppato da Google DeepMind, **SynthID** rappresenta uno dei strumenti di riferimento per il watermarking invisibile di immagini generate [12, 13]. Integrata nativamente nelle pipeline di inferenza di modelli quali *Imagen* e *Gemini*, questa tecnologia bypassa la fragilità dei metadati tradizionali iniettando la firma matematica direttamente nella micro-struttura dell'immagine.

Architettura e Meccanismo di Inserimento

Il framework di SynthID è composto da due modelli neurali profondi addestrati congiuntamente con obiettivi parzialmente contrapposti:

- **L'Embedder (E):** Interviene durante la fase di sintesi dell'immagine. Operando una trasformazione nel dominio spettrale, seleziona specifici coefficienti spettrali su cui iniettare il payload informativo. L'ottimizzazione dell'Embedder è vincolata da una loss di percettibilità che impone all'immagine marcata $I_w = I + \delta$ di essere visivamente indistinguibile dall'originale I.

- **Il Detector (D):** Un modello di classificazione associato che riceve un'immagine incognita I^* e calcola una mappa di confidenza $D(I^*) \in [0, 1]$, indicante la probabilità di presenza della firma di SynthID.

Durante il processo di *robustness training*, l'immagine prodotta dall'Embedder viene sottoposta a una pipeline di attacchi simulati prima di essere analizzata dal Detector. In caso di mancato rilevamento, la loss complessiva penalizza l'Embedder, costringendolo a distribuire il segnale in maniera ridondante sull'intera struttura spettrale dell'immagine, aumentandone la robustezza a trasformazioni geometriche e fotometriche.

Analisi dei Vantaggi e dei Limiti

Il principale punto di forza di SynthID risiede nella sua **robustezza al Cropping**, garantita dalla distribuzione spazialmente uniforme del segnale sull'intera struttura dell'immagine. Poiché il watermark non è localizzato in una regione specifica ma diffuso su tutti i coefficienti spettrali, anche porzioni ridotte dell'immagine originale sono sufficienti per permettere al Detector di superare la soglia di rilevamento. Inoltre, l'integrazione nella pipeline di generazione garantisce una protezione nativa, insensibile alle conversioni di formato e alle compressioni lossy tipiche dei canali di distribuzione digitale.

Di contro, il sistema manifesta una totale dipendenza da infrastrutture proprietarie. Il modello di rilevamento non è open-source, configurando SynthID come un sistema di verifica chiuso e binario: esso non è in grado di determinare se un'immagine sia stata generata da un modello generativo generico, ma può esclusivamente attestare l'appartenenza della firma all'ecosistema Google.

Modelli di Attacco e Vulnerabilità

La robustezza di *SynthID* è stata oggetto di approfondite verifiche di sicurezza e tentativi di *reverse-engineering*, i quali hanno evidenziato tre potenziali vettori di vulnerabilità:

1. **Texture Confusion:** Si evidenzia una fluttuazione dell'accuratezza quando il framework viene applicato a immagini caratterizzate da texture ad altissima frequenza e pattern complessi, quali fogliame denso o pellicce animali [13]. In questi scenari, gli studi suggeriscono che le componenti spettrali ad alta frequenza naturalmente presenti nel contenuto visivo tendono a sovrapporsi alle bande frequenziali teoricamente allocate dall'encoder; questo fenomeno agisce come una distorsione naturale latente, che può complicare la discriminazione dei pattern da parte del Detector.

2. **Neutralizzazione Spettrale via Analisi di Fourier:** Secondo studi di terze parti focalizzati sulla decodifica del segnale, i tentativi di *reverse-engineering* hanno evidenziato la presenza di picchi di magnitudo ricorrenti nel dominio delle frequenze [14]. I ricercatori hanno quindi ipotizzato che l'al-

goritmo possa basarsi sull'impiego di frequenze portanti fisse con fase costante nello spazio spettrale. Basandosi su questo modello teorico, l'applicazione della Trasformata di Fourier Bidimensionale ($2D\text{-}FFT$) su ampi dataset di immagini marcate ha permesso agli analisti di individuare anomalie statistiche in precise coordinate frequenziali, suggerendo la fattibilità di attacchi di filtraggio mirati capaci di compromettere l'estrazione del watermark senza degradare la qualità visiva dell'asset.

3. **Attacchi di Rigenerazione:** Questo attacco, noto come *diffusion purification* [14], mira alla rimozione della firma tramite una parziale risintesi dell'immagine. Data l'immagine marcata I_w, l'attaccante applica un processo di diffusione in avanti per un numero ridotto di passi temporali t, introducendo una quantità controllata di rumore gaussiano; un modello di diffusione esegue poi la traiettoria di denoising inversa per ricostruire l'immagine. Poiché il watermark risiede nelle componenti ad alta frequenza, particolarmente sensibili alle perturbazioni stocastiche introdotte dal processo di diffusione, questo meccanismo tende a degradare il segnale invisibile pur preservando la semantica visiva dell'immagine originale.

1.3.2 Meta: Pixel Seal & Meta Seal

La risposta di Meta al problema del tracciamento si è concretizzata nel rilascio di framework open-source orientati alla comunità scientifica, culminando nell'introduzione di **Pixel Seal** [15], i quali si propongono come evoluzione di approcci precedenti come *Stable Signature*.

Meccanismo Tecnico dell'Adversarial-Only Training

Il funzionamento di Pixel Seal supera il classico paradigma di addestramento degli autoencoder steganografici. Nella letteratura precedente, la preservazione dell'invisibilità visiva veniva imposta forzando la rete tramite metriche matematiche esplicite di distanza spaziale o percettiva (MSE, $SSIM$, $LPIPS$). I ricercatori di Meta hanno dimostrato che tali vincoli rigidi limitano artificialmente la capacità di embedding della rete, riducendo la robustezza del segnale [15].

Pixel Seal implementa un addestramento di tipo **Adversarial-Only**. Un modello discriminatore viene addestrato congiuntamente all'Embedder tramite una loss *adversarial*: l'Embedder è ottimizzato affinché l'immagine marcata risulti indistinguibile dall'originale secondo il discriminatore, imponendo un vincolo implicito sulla distribuzione delle perturbazioni introdotte. Questo schema incentiva l'Embedder a sviluppare strategie di occultamento del payload che operano in regioni dello spazio percettivo difficilmente rilevabili dall'occhio umano, senza ricorrere a metriche esplicite di distanza pixel-wise. Il payload iniettato è tipicamente un messaggio binario robusto $M \in \{0,1\}^L$ (con $L = 256$ bit). Durante il training, la stabilità rispetto alle manipolazioni viene ottenuta inserendo un simulatore di attacchi differenziabile tra l'Embedder e l'Estrattore, minimizzando la *Message Loss*, definita come la binary cross-entropy bit per bit tra il messaggio originale M e il messaggio stimato $\hat{M}$ prodotto dall'Estrattore:

$$\mathcal{L}_{\text{msg}} = -\frac{1}{L}\sum_{i=1}^{L}\Big[M_i \log(\hat{M}_i) + (1 - M_i)\log(1 - \hat{M}_i)\Big] \tag{1.1}$$

dove $M_i \in \{0,1\}$ è l'i-esimo bit del messaggio originale e $\hat{M}_i \in [0,1]$ è la probabilità predetta dall'Estrattore che il bit estratto sia 1. La minimizzazione di $\mathcal{L}_{\text{msg}}$ spinge l'Estrattore a recuperare fedelmente il payload anche in presenza delle distorsioni simulate.

Vantaggi e Criticità Operative

L'approccio Adversarial-Only permette a Pixel Seal di posizionarsi allo stato dell'arte [15] in termini di robustezza operativa, resistendo a compressioni web distruttive e screenshot ripetuti. Nello specifico, nel lavoro di Souček e colleghi [15], la valutazione viene condotta su un insieme di 1000 immagini ad alta risoluzione, composte da immagini sintetiche generate da modelli generativi e da fotografie reali tratte dal dataset SA-1b [16], confrontando Pixel Seal con baseline multi-bit consolidate quali MBRS, TrustMark, InvisMark e WAM. La tolleranza alle compressioni web distruttive viene modellata applicando una compressione JPEG con fattore di qualità pari a 80, atta a simulare i processi più comuni di codifica *lossy* della rete. La resistenza agli screenshot e alle manipolazioni editoriali viene invece

validata simulando sia operazioni di ritaglio, con un *center crop* del 10% con successivo ridimensionamento alle dimensioni originarie di 1024×1024 pixel, sia una pipeline di degradazione cumulativa tipica dei cicli di condivisione sui *social media*; quest'ultima prevede un downscaling al 75% della risoluzione, una compressione JPEG aggressiva a qualità 70 e un finale upscaling per ripristinare la risoluzione nativa dell'asset. Inoltre, configurandosi come una tecnologia *Post-Hoc*, può essere applicata direttamente sulle mappe di pixel di immagini preesistenti, rendendola universale e indipendente dal modello generativo a monte.

La criticità principale risiede nella natura intrinseca dell'addestramento *adversarial*: se il discriminatore presenta regioni dello spazio percettivo non adeguatamente coperte durante il training (*blind spots*), l'Embedder tenderà a sfruttarle, generando artefatti visivi localizzati o anomalie cromatiche imprevedibili. Inoltre, la capacità di rilevamento è soggetta a un effetto soglia: la *bit accuracy* dell'Estrattore, ovvero la frazione di bit del payload recuperati correttamente, deve superare il 75% affinché il watermark sia considerato integro. Al di sotto di tale soglia, il segnale estratto non è sufficientemente affidabile da consentire l'attribuzione, e il watermark viene considerato compromesso.

Analisi delle Vulnerabilità

Nonostante le elevate performance, Pixel Seal mostra vulnerabilità di fronte ad attacchi geometrici di desincronizzazione [15]. Operazioni quali distorsioni prospettiche non lineari o suddivisioni dell'immagine in porzioni non contigue compromettono l'allineamento spaziale necessario all'Estrattore per campionare correttamente la struttura dell'immagine, degradando la bit accuracy al di sotto della soglia di rilevamento. Questo limite è intrinseco alla dipendenza del modello dalla struttura spaziale dell'immagine originale: l'Estrattore, basato su un'architettura densamente convoluzionale, è ottimizzato per operare su immagini geometricamente coerenti con quelle viste durante il training.

Un ulteriore profilo di vulnerabilità, di natura architetturale, è formalizzato da Nemecek e colleghi [17] sotto il nome di *Integrity Clash*. Il fenomeno origina dalla separazione tecnica tra due livelli di verifica dell'autenticità dei contenuti digitali: gli standard crittografici di provenienza, quali C2PA, che operano tramite manife-

sti firmati digitalmente, e il watermarking invisibile a livello di pixel. Poiché i due livelli di verifica sono tecnicamente indipendenti, un attaccante può manipolare un'immagine AI-generated inserendo un manifesto C2PA contraffatto che ne attribuisce la creazione a un autore umano, lasciando intatto il watermark invisibile nei pixel. In fase di analisi, il verificatore C2PA restituirà esito positivo sulla provenienza dichiarata, e il detector del watermark restituirà esito positivo sull'origine AI: due segnali autenticati ma semanticamente opposti, senza che nessuno dei due sistemi sia in grado di rilevare la contraddizione.

1.3.3 OpenAI: C2PA e Metadati di Provenienza

In netta controtendenza rispetto all'approccio basato sulla modifica dei pixel e sulla marcatura del segnale visivo, OpenAI ha strutturato la sicurezza dei propri modelli di punta, come *DALL-E 3*, attorno al concetto di *provenienza crittografica*, implementando nativamente le specifiche della **Coalition for Content Provenance and Authenticity (C2PA)** [18].

Architettura e Catena di Custodia

Lo standard C2PA non altera la componente visiva del file multimediale, ma introduce una struttura logica standardizzata denominata *C2PA Manifest* all'interno dei blocchi metadati JUMBF (*JPEG Universal Metadata Box Format*) del file [19]. JUMBF è un *container format* standardizzato da ISO/IEC che consente di incorporare strutture di metadati arbitrarie e tipizzate direttamente nel flusso binario di un file multimediale, senza modificarne il contenuto visivo o il comportamento applicativo. Il processo si articola in tre fasi distinte:

1. **Generazione e Hashing:** Nel momento in cui il modello sintetizza l'immagine, l'infrastruttura cloud calcola un hash crittografico sicuro (tipicamente SHA-256) del contenuto codificato del file [18].

2. **Firma Digitale:** L'hash calcolato, unitamente alle asserzioni di provenienza (*assertions*) che attestano la data di generazione e l'identità del modello generativo, viene inserito nel manifesto. Questo blocco viene firmato digi-

talmente utilizzando una chiave crittografica privata di OpenAI ancorata a una Public Key Infrastructure (PKI) riconosciuta [19].

3. **Catena di Custodia (Chain of Custody):** Se l'immagine viene successivamente modificata tramite strumenti conformi, il sistema applica una struttura ricorsiva: non sovrascrive il manifesto originale, ma concatena un nuovo manifesto contenente l'hash della nuova immagine e le credenziali dell'editor, preservando l'intera genealogia del file [19].

Vantaggi e Limiti Strutturali

I vantaggi risiedono nell'interoperabilità globale garantita dallo standard aperto e nell'elevata affidabilità crittografica dei processi di autenticazione dell'origine. Finché l'integrità del box JUMBF e la firma crittografica risultano intatte, l'origine dell'immagine è verificabile in modo univoco. Il limite fondamentale risiede tuttavia nell'estrema **fragilità strutturale** del trasporto dei metadati: poiché il manifesto è ospitato all'interno del contenitore logico del file e non nei suoi pixel, esso è soggetto a rimozione da parte di numerosi servizi web e piattaforme di distribuzione, che eliminano i metadati per ragioni di compressione o privacy (*metadata stripping*) [19].

Vettori di Bypass Tecnico e Desincronizzazione

Le specifiche tecniche e la letteratura recente evidenziano la vulnerabilità intrinseca del sistema nei confronti di operazioni di slegamento (*stripping*) tra il manifesto e il contenuto visivo, che determinano scenari di totale desincronizzazione della provenienza:

- **Screenshot Attack:** La cattura dello schermo produce una nuova acquisizione raster della rappresentazione visiva renderizzata, generando un file completamente indipendente dall'originale. Questo processo esclude intrinsecamente i segmenti JUMBF d'origine, interrompendo la catena di custodia e lasciando l'immagine priva di credenziali di provenienza, pur conservandone la semantica visiva [17].

- **Stripping Strumentale:** L'utilizzo di software di elaborazione delle immagini che non preservano i blocchi di metadati durante il salvataggio consente la rimozione diretta del manifesto C2PA [18]. Il file risultante rimane visivamente identico all'originale, ma perde il riferimento crittografico alla firma del provider, rendendo impossibile verificarne la provenienza attraverso i canali C2PA.

Evoluzione dell'Infrastruttura di Provenienza in OpenAI

Al fine di strutturare un ecosistema di fiducia multilivello, OpenAI ha esteso la propria architettura di sicurezza integrando nei flussi di ChatGPT, Codex e delle proprie API la tecnologia di watermarking *SynthID* di Google (Figura 1.7)[20]. Questa convergenza implementa una strategia di difesa in profondità che mappa il segnale su due piani distinti: all'approccio dichiarativo e contestuale dello standard aperto Coalition for Content Provenance and Authenticity (C2PA) si affianca una marcatura spettrale invisibile a livello di pixel [20]. Il principio cardine risiede nella complementarietà dei due layer: laddove il manifesto crittografico C2PA veicola i metadati storici dettagliati dell'asset, l'algoritmo di *SynthID* garantisce la persistenza e il recupero della firma qualora i metadati vengano rimossi, resistendo a conversioni di formato e screenshot software [20].

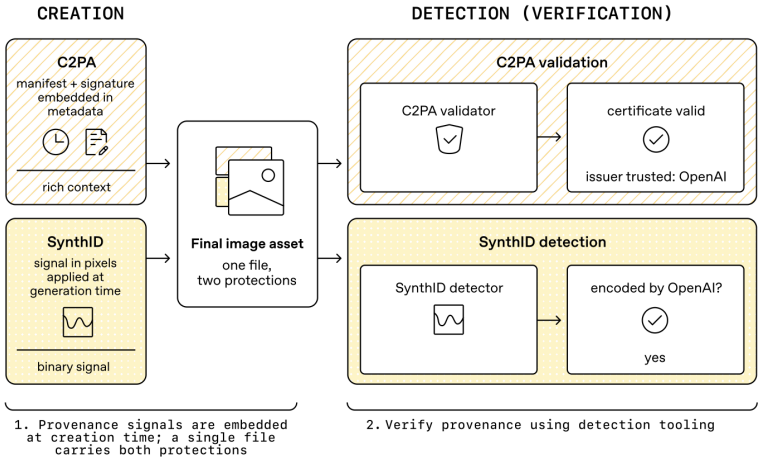


Figura 1.7: Rappresentazione concettuale del protocollo di autenticazione cross-layer di OpenAI, basato sull'ispezione simultanea del watermark invisibile SynthID (livello pixel) e dei Content Credentials C2PA (livello metadati).

1.3.4 L'Ecosistema Microsoft: Implementazione Enterprise di C2PA e Content Credentials

Microsoft ha adottato gli standard aperti C2PA come riferimento tecnico per la tracciabilità dei contenuti multimediali, formalizzando l'integrazione delle Content Credentials all'interno del programma LASER (*Longer-term AI Safety in Engineering and Research*), coordinato dall'Ufficio del Chief Scientific Officer [21]. In questo contesto, il contributo di Microsoft si focalizza sulla traduzione delle specifiche C2PA in implementazioni concrete all'interno dei propri sistemi di generazione e distribuzione di contenuti.

Integrazione e Ibridazione del Segnale (Layering Tecnologico)

Per mitigare i limiti intrinseci dei singoli meccanismi di autenticazione, il report di Young e colleghi [21] descrive un modello di protezione multi-livello che combina provenienza crittografica, watermarking imperceibile e fingerprinting. In questa architettura, il manifesto C2PA non opera in isolamento, ma viene affiancato da soluzioni di marcatura imperceibile quali *InvisMark* [22], con il watermark che assolve la funzione di rendere più difficile la rimozione delle credenziali di provenienza.

Questo approccio multi-livello mira a risolvere la vulnerabilità strutturale derivante dall'indipendenza tecnica tra il piano dei metadati crittografici e quello dei pixel [17]. L'integrazione di un watermark robusto consente di preservare un canale alternativo di verifica della provenienza anche qualora il manifesto C2PA venga rimosso o cancellato durante la distribuzione, ad esempio per effetto di metadata stripping da parte di piattaforme non conformi o di software di elaborazione delle immagini che non ne preservano i blocchi di metadati [17, 22].

Vulnerabilità di Settore: Attacchi di Rigenerazione e Limiti del Soft-Hashing

I protocolli di tracciabilità basati su fingerprinting e i sistemi di certificazione locale mostrano tuttavia una vulnerabilità generalizzata nei confronti dei cosiddetti **attacchi di rigenerazione** (*re-generation attacks*), analizzati come uno dei vettori più complessi per il mantenimento dell'integrità del segnale di provenienza

[21]. Come illustrato in Figura 1.8, per ovviare alla volatilità dei metadati C2PA, i principali attori dell'ecosistema di autenticazione dei contenuti, tra cui i membri della Content Authenticity Initiative come Adobe, Google e Microsoft, si affidano a funzioni di *soft-hashing* percettivo. Queste funzioni generano rappresentazioni compatte dell'asset che preservano la similarità percettiva tra immagini visivamente affini, proiettando le informazioni visive in descrittori stabili attraverso tre approcci principali: scomposizione in blocchi spaziali con estrazione di statistiche locali, analisi frequenziale tramite trasformate discrete, e hashing percettivo binario tramite proiezioni a bassa dimensionalità, al fine di consentire un tracciamento indicizzato all'interno di database di provenienza centralizzati [21].

Figura 1.8: Rappresentazione concettuale delle tre metodologie generali di *soft-hashing* analizzate nel report di Microsoft Research [21].

Se un attore avversario sottopone l'asset originale firmato a un secondo modello generativo, applicando la *diffusion purification*, una tecnica originariamente sviluppata nel contesto della robustezza *adversarial* e successivamente impiegata come vettore di attacco ai sistemi di provenienza, il modello ricostruisce la mappa dei pixel campionando la struttura semantica del file sorgente attraverso un ciclo di forward noising parziale e denoising neurale. Questa operazione altera le statistiche locali dei coefficienti frequenziali e la distribuzione spaziale dei descrittori estratti per blocco su cui si basano le funzioni di soft-hashing. Di conseguenza, il processo rimuove le perturbazioni caratteristiche degli eventuali watermark invisibili e invalida i segmenti del manifesto C2PA originario, producendo un file la cui firma percettiva non corrisponde ad alcun asset registrato nel database di riferimento, o la cui firma risulta consistente con quella dell'asset originale ma priva del manifesto di autenticità associato. Quest'ultimo scenario, in cui il descrittore

percettivo indica una corrispondenza con contenuto certificato mentre i metadati di provenienza risultano assenti o alterati, costituisce un caso concreto di Integrity Clash nel senso formalizzato da Nemecek et al. [17, 21]: due segnali autenticati ma semanticamente opposti, senza che nessuno dei due sistemi sia in grado di rilevare la contraddizione.

Capitolo 2

Metodologia Sperimentale e Tecniche di Rimozione

2.1 Analisi tecnica dei sistemi di rimozione del watermark

Nell'ambito del watermarking basato su reti neurali, la robustezza di un sistema si valuta rispetto a un insieme di trasformazioni che l'immagine con watermark può subire lungo il suo ciclo di vita. Oltre ai tentativi deliberati di rimozione, nella valutazione della robustezza si considerano comunemente anche processi non intenzionali, come la ricompressione e il ridimensionamento operati dalle piattaforme di condivisione (quali Instagram, WhatsApp o X), che possono compromettere il recupero del watermark. Con tale espressione si intende qui sia la perdita di *rilevabilità* della firma, ovvero sia l'incapacità del detector di segnalarne la presenza, sia la corruzione del *payload*, cioè degli effettivi bit del messaggio incorporato.

Tali perturbazioni agiscono alterando la distribuzione dei valori dei pixel nel dominio spaziale, attraverso manipolazioni geometriche quali crop e resize, oppure mediante compressioni lossy. Affinché l'attacco sia significativo, tale degradazione non deve però intaccare il contenuto visibile dell'immagine, ossia ciò che ne percepisce l'osservatore umano: la perturbazione deve preservarne sia la fedeltà percettiva, cioè l'assenza di artefatti visibili rispetto all'originale, sia il contenuto semantico, vale a dire gli oggetti e la scena rappresentati. Un'alterazione che

degradi sensibilmente la qualità dell'immagine, infatti, non costituisce un attacco utile, poiché rende evidente la manipolazione e compromette il valore stesso del contenuto.

L'efficacia di un attacco si valuta dunque secondo un duplice criterio. Da un lato, la capacità di degradare il segnale del watermark al di sotto della soglia oltre la quale il detector non riesce più a recuperare l'informazione incorporata in modo affidabile, che si tratti della decodifica di un messaggio binario, nel caso di sistemi con payload, o del rilevamento della sola presenza della firma, nel caso di sistemi basati esclusivamente su un test di ipotesi. Dall'altro, la conservazione del contenuto visivo originale, quantificabile mediante metriche di distorsione quali il Peak Signal-to-Noise Ratio (PSNR) o, in modo più aderente alla percezione umana, metriche percettive quali Structural Similarity Index Measure (SSIM) e Learned Perceptual Image Patch Similarity (LPIPS). Il successo dell'attacco non coincide quindi con la distruzione dell'immagine, bensì con il fallimento del meccanismo di verifica del watermark, mantenendo al contempo una qualità visiva comparabile a quella dell'immagine di partenza.

2.2 Dataset sperimentale

Il materiale di partenza è costituito da 100 *triplette* di immagini (Figura 2.1). Ogni tripletta è composta da un'immagine reale e da due immagini sintetiche da essa derivate:

- un'immagine **reale**: immagini reali di riferimento, prive di watermark, utilizzate come ground truth per il termine di fedeltà ricostruttiva.
- una versione **text-to-image (T2I)**: immagini sintetiche generate a partire dal solo prompt testuale.
- una versione **image+text-to-image (IT2I)**: immagini sintetiche generate condizionando il modello su un'immagine di riferimento congiuntamente a un prompt testuale.

Il prompt impiegato per la generazione è ottenuto descrivendo automaticamente ciascuna immagine reale con il modello vision-language *Qwen3.5* [23] e verrà

utilizzato per la generazione delle due controparti sintetiche tramite *NanoBanana* (Google), che incorpora nativamente il watermark SynthID al momento della generazione. Le tre versioni rappresentano così il medesimo contenuto, ma con origine e modalità di generazione differenti.

Un'osservazione rilevante riguarda la categoria IT2I: poiché la generazione è condizionata sull'immagine reale di partenza, le immagini risultanti tendono a preservarne la composizione e i soggetti principali, risultando spesso visivamente più simili all'originale rispetto alle controparti T2I. Questa somiglianza ha implicazioni dirette sull'attacco a SynthID: le IT2I condividono con le reali elementi strutturali e cromatici che potrebbero influenzare la coerenza spettrale del watermark, differenziandole dalle T2I generate da zero su una risoluzione nativa del modello.

2.2.1 Analisi della composizione del dataset

Il dataset è composto in modo da rappresentare soggetti eterogenei, spaziando tra ambiti differenti, tra cui scene urbane, ritratti, animali, cibo, paesaggi e oggetti, così da garantire una varietà visiva ampia e bilanciata. La Tabella 2.1 ne riassume la composizione: le immagini reali sono prive di watermark, mentre le due categorie sintetiche integrano il watermark SynthID in fase di generazione.

Tabella 2.1: Composizione del dataset sperimentale. Le risoluzioni sono espresse in pixel; l'orientamento riporta il numero di immagini orizzontali, verticali e quadrate.

Categoria	**N. img**	**Ris. distinte**	**Megapixel**	**Orient.**	**Watermark**
`real`	100	34	0,55–1,05	71/23/6	assente
`text_to_image`	100	3	1,06–1,08	100/0/0	presente
`image_text_to_image`	100	24	1,05–1,08	77/18/5	presente
Totale	**300**	—	—	—	—

Le due categorie sintetiche presentano caratteristiche di risoluzione marcatamente diverse. Le immagini T2I mostrano una risoluzione sostanzialmente costante, corrispondente alla risoluzione nativa del generatore **nano-banana-pro-preview**, come si evince dal numero ridotto di risoluzioni distinte riportato in Tabella 2.1. Le IT2I ereditano invece risoluzione e proporzioni dall'immagine rea-

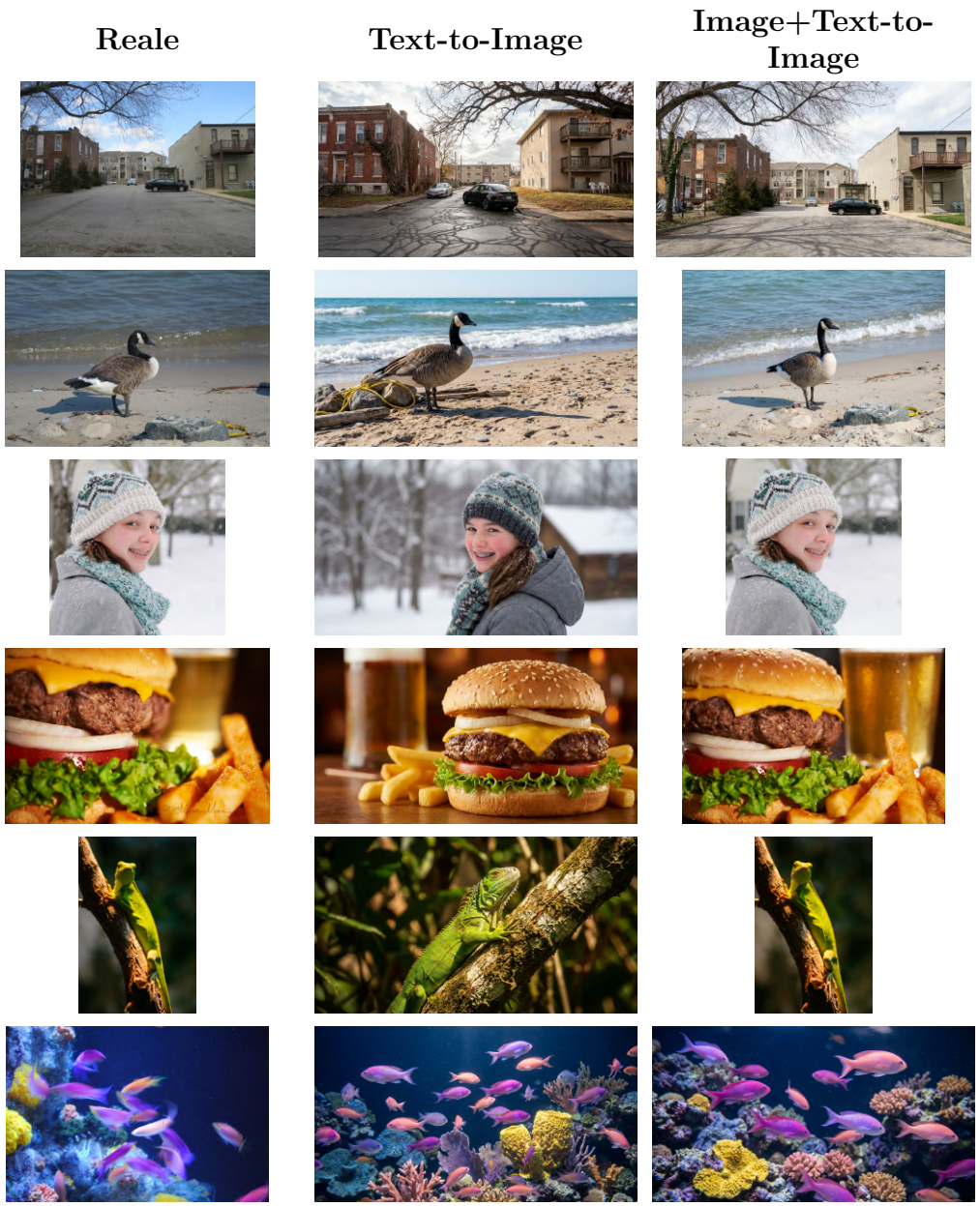


Figura 2.1: Esempi di triplette del dataset. Ogni riga mostra una stessa scena nelle tre categorie: l'immagine reale di riferimento (priva di watermark), la controparte *text-to-image* generata dal solo prompt testuale, e la versione *image+text-to-image* generata condizionando il modello sull'immagine reale e sul medesimo prompt, entrambe recanti il watermark SynthID di NanoBanana.

le di condizionamento, risultando molto più eterogenee. Poiché, secondo le analisi condotte dagli autori di reverse-SynthID, le frequenze portanti di SynthID dipendono dalla risoluzione dell'immagine (Sezione 2.3.1), i watermark presenti nelle due categorie possono essere associati a profili spettrali differenti.

2.3 SynthID

2.3.1 Reverse-Engineering di SynthID mediante analisi spettrale

SynthID è il sistema di watermarking invisibile sviluppato da Google DeepMind, incorporato nelle immagini generate dai sistemi Google che adottano SynthID [12]. Il watermark non è percettibile all'occhio umano, ma è rilevabile algoritmicamente e consente a Google il rilevamento della provenienza artificiale dell'immagine.

Il progetto *reverse-SynthID* [24] si propone di analizzare, rilevare e rimuovere tale watermark (Figura 2.2) impiegando unicamente tecniche di elaborazione del segnale, senza alcun accesso all'encoder o al decoder proprietario di Google, attraverso analisi spettrale.

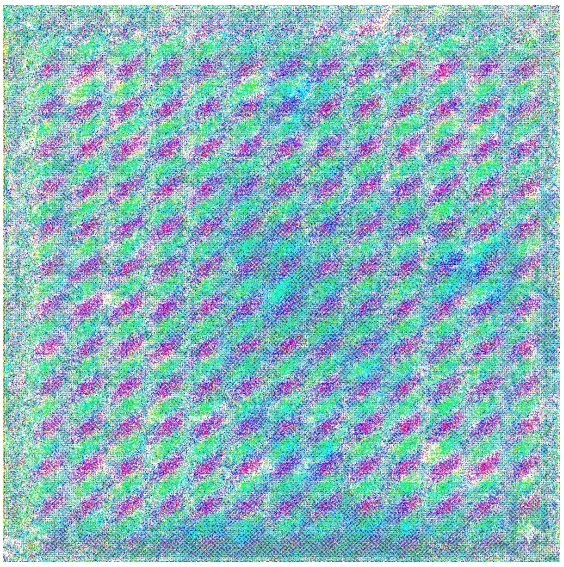


Figura 2.2: Pattern portante SynthID amplificato estratto da un'immagine Gemini completamente bianca. La banda diagonale è interpretata dagli autori come la firma di frequenza spaziale del watermark, bersaglio dell'attacco spettrale. [24]

Funzionamento dell'encoder SynthID

Gli autori del progetto ipotizzano che il meccanismo di embedding possa essere descritto mediante due componenti principali (Figura 2.3).

- **Encoder**: Vengono selezionate frequenze portanti (*carrier frequencies*), ossia specifiche componenti dello spettro di Fourier bidimensionale dell'immagine, ciascuna individuata da una coppia di coordinate (f_y, f_x) e utilizzata per codificare il segnale del watermark. Secondo questa ricostruzione, a ciascuna frequenza portante sarebbe assegnato un valore di fase *fisso*; gli autori osservano infatti un'elevata coerenza di fase in corrispondenza delle portanti tra immagini diverse generate dallo stesso modello (Sezione 2.3.1), compatibile con l'ipotesi di un template di fase condiviso. Il comportamento osservato è inoltre compatibile con la presenza di un encoder neurale che sovrappone all'immagine un pattern di rumore appreso, distribuito sull'intero spettro in modo da risultare impercettibile.

- **Decoder (analisi degli autori)**: Il blocco di rilevamento qui descritto non è il decoder proprietario di Google, bensì la procedura di analisi sviluppata nel progetto reverse-SynthID per individuare il watermark SynthID. A partire dall'immagine si stima il *residuo di rumore*, ovvero la componente ad alta frequenza che rimane sottraendo all'immagine una sua versione regolarizzata, in cui è concentrato il segnale del watermark. Su tale residuo si applica la Trasformata di Fourier (FFT) e si verifica la fase alle frequenze portanti attese: se essa risulta coerente con il template ricostruito dagli autori, l'analisi segnala la presenza del watermark SynthID nell'immagine.

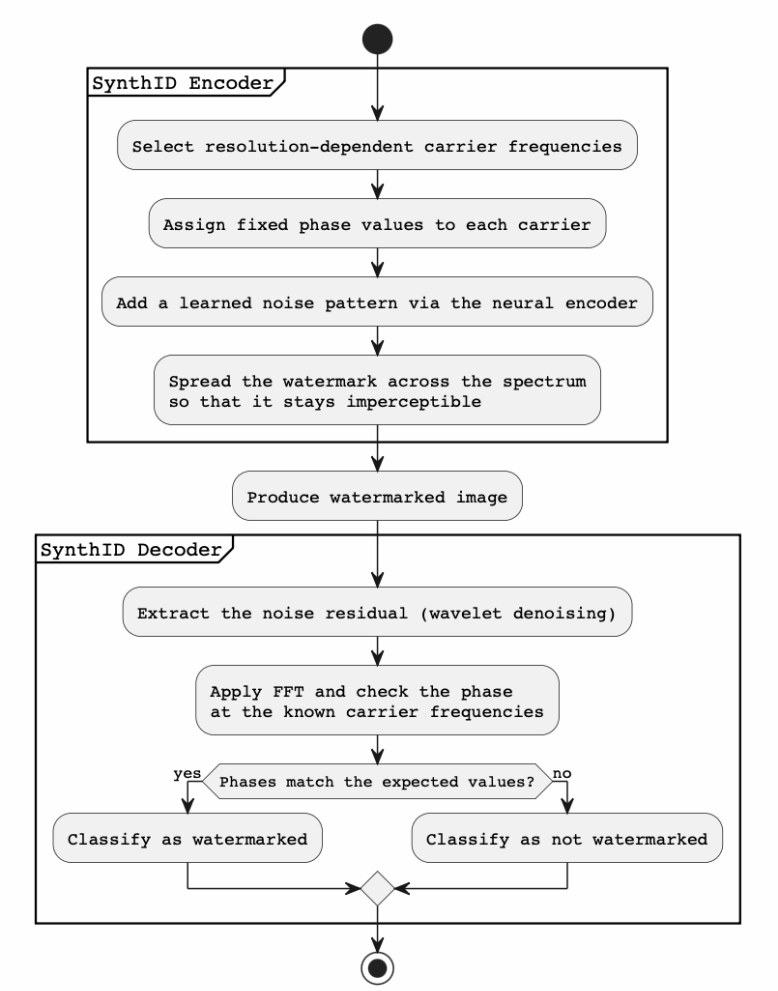


Figura 2.3: Pipeline del sistema di watermarking SynthID ricostruita tramite reverse engineering.

Uno dei risultati principali dell'analisi riguarda la dipendenza dalla risoluzione delle frequenze portanti del watermark. Gli autori osservano che le coordinate (f_y, f_x) in cui il segnale del watermark si concentra nello spettro di Fourier non sono fisse, ma cambiano al variare della risoluzione dell'immagine: una stessa frequenza portante occupa posizioni differenti nello spettro a seconda delle dimensioni in pixel dell'immagine analizzata. Di conseguenza, i parametri di rilevamento calibrati su una risoluzione non sono trasferibili a un'altra. Questi parametri sono raccolti in una struttura dati interna al progetto reverse-SynthID, denominata *codebook*, che archivia per ciascun profilo di risoluzione le posizioni delle frequenze portanti,

le magnitudini stimate e i valori di fase del watermark: un codebook calibrato su immagini 1024×1024 non risulta quindi applicabile a immagini 1536×2816, poiché le frequenze portanti vi occupano componenti spettrali completamente differenti.

L'analisi condotta dagli autori suggerisce l'esistenza di un template di fase altamente consistente tra immagini generate dallo stesso modello Gemini:

- Il canale verde (G) mostra la maggiore energia associata al watermark nell'analisi condotta dagli autori;

- La coerenza di fase media cross-immagine alle frequenze portanti selezionate supera il 99.5%. Per coerenza di fase si intende il grado con cui il valore di fase misurato a una data frequenza portante si mantiene costante al variare dell'immagine: indicando con ϕ_i la fase osservata nell'immagine i-esima, essa si quantifica come il modulo della media dei vettori di fase $e^{j\phi_i}$,

$$R = \left| \frac{1}{N} \sum_{i=1}^{N} e^{j\phi_i} \right| \in [0, 1]$$

  dove N è il numero di immagini considerate; un valore prossimo a 1 indica che le fasi sono quasi identiche tra le immagini, compatibilmente con la presenza di una componente deterministica quale un watermark, mentre un valore prossimo a 0 indica fasi distribuite in modo pressoché casuale, come ci si attende per contenuto visivo non correlato;

- La verifica differenziale tra immagini di riferimento cromaticamente uniformi, generate interamente a valore di pixel costante, rispettivamente nero e bianco, consente di separare le frequenze portanti attribuibili al watermark da quelle dovute a componenti spettrali sistematiche introdotte dal processo di generazione, presenti indipendentemente dal watermark. Lo strumento impiegato è la misura di allineamento di fase $|\cos(\Delta\phi)|$, dove $\Delta\phi = \phi_{\text{oss}} - \phi_{\text{att}}$ è la differenza tra la fase osservata ϕ_{oss} a una data frequenza portante e la fase attesa ϕ_{att} prevista dal template ricostruito. Il valore $|\cos(\Delta\phi)|$ è prossimo a 1 quando le due fasi sono identiche o opposte, condizione compatibile con un segnale di watermark deterministico, e prossimo a 0 quando non lo sono. Una frequenza portante è ritenuta attribuibile al watermark dagli autori

quando $|\cos(\Delta\phi)| > 0.90$, soglia determinata empiricamente nel progetto reverse-SynthID.

Struttura del codebook multi-risoluzione

Il *SpectralCodebook*, introdotto dagli autori di reverse-SynthID, è il componente centrale del sistema e costruisce la base operativa di tutte le pipeline di bypass descritte nelle sezioni successive. Viene costruito a partire da due tipologie di immagini di riferimento:

- **Immagini a tinta unita (bianche e nere)** (risoluzione 1024×1024): sfondi privi di contenuto semantico che minimizzano l'influenza del contenuto visivo sullo spettro, rendendo il segnale del watermark più facilmente osservabile; vengono utilizzati come riferimento spettrale per l'identificazione delle frequenze portanti.
- **Immagini con watermark e contenuto visivo** (risoluzione 1536×2816): il contributo del contenuto visivo tende a cancellarsi mediante media statistica su un insieme numeroso di immagini diverse, lasciando emergere la componente del watermark nel dominio spettrale. La magnitudine stimata del watermark a ciascuna componente frequenziale discreta è data da

$$\hat{A}[f] = \bar{A}[f] \cdot c[f]^2$$

dove f individua la componente frequenziale discreta, determinata dalle coordinate (f_y, f_x) nella griglia dello spettro di Fourier bidimensionale; $\hat{A}[f]$ è la magnitudine del watermark stimata a tale componente, ovvero l'ampiezza del segnale che si intende ricostruire; $\bar{A}[f]$ è la magnitudine media osservata a quella componente sull'insieme delle immagini, dalla quale il contenuto visivo si è cancellato per effetto della media statistica, lasciando prevalere la componente di watermark; infine $c[f]$ è il coefficiente di confidenza associato alla componente, un valore compreso tra 0 e 1 che quantifica l'affidabilità della stima ed entra nella formula elevato al quadrato, così da pesare maggiormente le componenti su cui la stima è più attendibile e attenuare quelle incerte.

Il codebook archivia, per ciascun profilo di risoluzione, le posizioni delle frequenze portanti, le magnitudini stimate, i valori di fase e i pesi di validazione. Al momento del bypass, `get_profile(H, W)` seleziona automaticamente il profilo di risoluzione esatta corrispondente all'immagine in ingresso o, in assenza di corrispondenza, quello di risoluzione più vicina disponibile.

Il codebook V4 estende il meccanismo introducendo un consenso di fase basato sull'aggregazione delle stime ottenute da immagini monocromatiche uniformi di sei colori distinti (nero, bianco, rosso, verde, blu, grigio) per ciascuna combinazione di modello generativo e risoluzione, organizzando 14 profili totali (7 risoluzioni $\times$ 2 modelli generativi: **gemini-3.1-flash-image-preview** e **nano-banana-pro-preview**) in una struttura indicizzata per `(modello, H, W)` [24].

Architettura del bypass: quattro generazioni

La qualità visiva delle immagini elaborate è quantificata mediante il PSNR, che misura il rapporto tra il segnale massimo e il rumore introdotto dall'elaborazione:

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

dove $MAX_I = 255$ per immagini RGB a 8 bit per canale e MSE è l'Errore Quadratico Medio tra l'immagine originale e quella elaborata. Valori superiori a 40 sono generalmente associati a differenze percettive ridotte.

Il progetto ha sviluppato quattro versioni successive di algoritmo di bypass, riassunte in Tabella 2.2.

Tabella 2.2: Confronto tra le quattro generazioni di algoritmo di bypass sviluppate nel progetto reverse-SynthID [24]. Il drop di coerenza di fase indica la riduzione percentuale della metrica R alle frequenze portanti rispetto al valore dell'immagine non attaccata: un valore elevato indica che il segnale del watermark è stato significativamente degradato.

Versione	Approccio	PSNR (dB)	Drop di coerenza di fase	Stato
V1	Compressione JPEG ($Q = 50$)	37	∼11%	Baseline
V2	Multi-stage (rumore, colore, frequenza)	27–37	∼0% (nessun effetto)	Scartato
V3	Sottrazione spettrale multi-risoluzione	43+	91%	Migliore precedente
V4	**Pipeline a 7 stadi (VAE + deformazione + ridimensionamento + colore + FFT + JPEG)**	—*	**Fallimento del rilevamento**	**Adottata**

* Metriche PSNR sistematiche non pubblicate nella release; la qualità visiva è stata valutata qualitativamente dagli autori come indistinguibile dall'originale.

Pipeline V3 (Multi-Resolution Spectral Bypass) La pipeline V3 opera come segue (Figura 2.4):

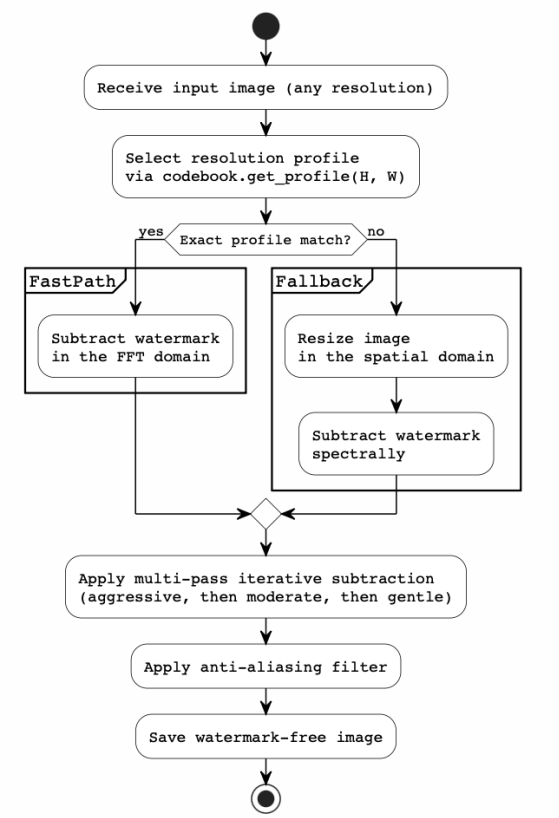


Figura 2.4: Schema della pipeline V3.

1. **Selezione del profilo**: `get_profile(H, W)` individua il profilo corrispondente alla risoluzione dell'immagine in ingresso; in caso di corrispondenza esatta si applica direttamente la sottrazione nel dominio FFT, altrimenti si ricorre a una procedura alternativa che prevede un ridimensionamento nel dominio spaziale prima di procedere con la sottrazione spettrale.

2. **Calcolo della confidenza**: per ogni frequenza portante, la confidenza c è definita come il prodotto di due fattori:

$$c = c_{\text{cof}} \times c_{\text{acc}},$$

dove c_{cof} è la coerenza di fase cross-immagine della frequenza portante (la metrica R già definita), e c_{acc} è il grado di accordo di fase tra sfondi monocromatici di colore diverso: poiché una vera frequenza portante SynthID è indipendente dal contenuto visivo, la sua fase risulta stabile al variare del colore di sfondo, mentre le componenti legate al contenuto mostrano fasi non coerenti tra un colore e l'altro.

3. **Sottrazione per bin**: indicando con $X[f]$ il coefficiente complesso dello spettro di Fourier dell'immagine alla componente f, la versione corretta dopo la rimozione del watermark è:

$$\hat{X}[f] = X[f] - w_{\text{ch}} \cdot A_{\text{wm}}[f] \cdot c[f] \cdot r_{\text{frac}}$$

dove $\hat{X}[f]$ è il coefficiente spettrale risultante dopo la sottrazione; $w_{\text{ch}} \in \{1.00;\ 0.85;\ 0.70\}$ è il peso associato al canale cromatico, con valori rispettivamente per i canali verde (G), rosso (R) e blu (B), che riflettono la diversa intensità con cui il watermark è incorporato nei tre canali; $A_{\text{wm}}[f]$ è la magnitudine stimata del watermark alla componente f, estratta dal codebook; $c[f]$ è il coefficiente di confidenza definito al punto precedente; e r_{frac} è la frazione di rimozione corrente, un parametro compreso tra 0 e 1 che controlla l'intensità della sottrazione nel passaggio corrente della procedura multi-pass.

4. **Vincolo di sottrazione massima**: la sottrazione è limitata in modo che non superi una soglia percentuale dell'energia del coefficiente alla componente frequenziale considerata; gli autori indicano un intervallo di riferimento compreso tra il 90% e il 95% [24], applicato per preservare la qualità visiva dell'immagine elaborata.

5. **Multi-pass decrescente**: La sequenza *aggressive* → *moderate* → *gentle* mira a ridurre progressivamente le componenti residue attribuite al watermark.

6. **Filtraggio anti-aliasing**: prima del salvataggio, viene applicato un filtro gaussiano con kernel 3×3 e deviazione standard $\sigma = 0.4$ nel dominio

spaziale, con l'obiettivo di attenuare le componenti ad alta frequenza che possono essere state introdotte o accentuate dalle operazioni di sottrazione spettrale nei passi precedenti, riducendo così potenziali artefatti visivi di campionamento [24].

Pipeline V4 (7-Stage All-in-One Bypass) La pipeline V4, rilasciata come pre-release il 23 aprile 2025 [24], rappresenta l'iterazione più recente del framework. A differenza della V3, che opera mediante sottrazione analitica nel dominio spettrale, la V4 implementa un attacco combinato non lineare a sette stadi sequenziali, finalizzati all'aggiramento del sistema di rilevamento (Figura 2.5):

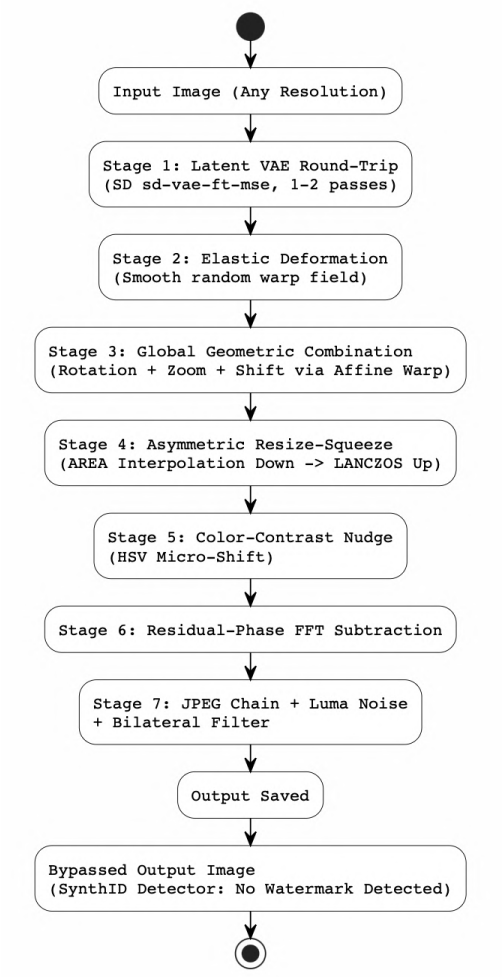


Figura 2.5: Flow della pipeline V4.

1. **Latent VAE Round-Trip**: L'immagine viene proiettata nello spazio latente di un Variational Autoencoder (VAE) e successivamente ricostruita. Questo passaggio tende ad attenuare alcune componenti ad alta frequenza in cui, secondo l'ipotesi degli autori, si ancora il pattern di rumore appreso dall'encoder neurale di SynthID.

2. **Elastic Deformation**: Applicazione di micro-spostamenti locali non lineari tramite campi di deformazione elastica a bassa frequenza. Questo stadio altera la distribuzione energetica associata ai bin frequenziali (f_y, f_x), riducendo la coerenza delle componenti frequenziali utilizzate per il rilevamento.

3. **Asymmetric Squeeze-Stretch**: Downscaling asimmetrico seguito da upscaling interpolato (es. $1024 \to 968 \to 1024$). L'interpolazione introduce errori di interpolazione e ricampionamento potenzialmente in grado di ridurre la coerenza di fase del watermark.

4. **Differential Color Blurring**: Secondo l'analisi degli autori, poiché il canale verde (G) veicola il segnale watermark più intenso, questo stadio applica un filtro gaussiano selettivamente su quel canale cromatico, attenuandone il contributo spettrale.

5. **Phase Noise Injection**: Iniezione di rumore sui coefficienti di fase delle frequenze portanti estratte dal codebook, con l'obiettivo di portare la metrica $|\cos(\Delta\phi)|$ verso zero e alterare il pattern di fase identificato dagli autori.

6. **Residual-Phase FFT Subtraction**: Sottrazione nel dominio frequenziale dei bin candidati associati al watermark nel codebook, analoga alla V3 ma applicata come stadio di raffinamento residuo.

7. **JPEG Chain + Luma Noise + Bilateral Filter**: il passaggio finale applica in sequenza tre operazioni. Una catena di compressioni JPEG a qualità decrescente ($Q = 92 \to 88$ nel preset `final`, $Q = 88 \to 84 \to 90$ nel preset `nuke`) introduce artefatti di quantizzazione nei coefficienti DCT, distruggendo la coerenza residua del watermark nello spettro frequenziale. A questi si aggiunge l'iniezione di rumore gaussiano sul canale di luminanza, che perturba le variazioni di intensità fine su cui il segnale del watermark

si ancora. Infine, un filtro bilaterale rimuove il rumore visibile introdotto dai passi precedenti preservando i contorni, riportando la qualità percettiva dell'immagine a un livello accettabile senza ripristinare la coerenza del watermark [24].

Il preset `final` corrisponde alla configurazione bilanciata dei parametri; il preset alternativo `nuke` applica la stessa sequenza a intensità massima [24].

Dataset di valutazione: immagini T2I e IT2I

L'attacco di rimozione del watermark SynthID è stato valutato su due categorie di immagini, per un totale di 200 immagini, sottoinsieme del dataset introdotto nella Sezione 2.2. Ciascuna categoria comprende 100 immagini generate dal modello **nano-banana-pro-preview**, che incorpora il watermark SynthID al momento stesso della generazione. Le due categorie si distinguono per la modalità di condizionamento del generatore:

- **Text-to-image (T2I)**: immagini sintetizzate a partire dal solo prompt testuale.
- **Image+text-to-image (IT2I)**: immagini generate condizionando il modello congiuntamente tramite un'immagine di riferimento e un prompt testuale.

Tale distinzione ha una conseguenza diretta sulla struttura spettrale del watermark, rilevante ai fini dell'attacco. Poiché, secondo le analisi condotte dagli autori di reverse-SynthID, le frequenze portanti di SynthID dipendono dalla risoluzione dell'immagine (Sezione 2.3.1), i watermark presenti nelle due categorie possono essere associati a profili spettrali differenti.

2.4 PixelSeal

2.4.1 Protocollo di valutazione e metriche

L'efficacia dell'attacco è quantificata confrontando, per ciascuna immagine, tre stati: *pulita*, *con watermark*, e *attaccata*. Per ogni stato si registrano:

- **Bit accuracy**: frazione di bit correttamente recuperati rispetto al messaggio incorporato nell'immagine con watermark di riferimento,

$$\text{bit\_acc} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{1}\left[\hat{b}_k = b_k^{wm}\right],$$

  dove $K = 256$ è il numero di bit del payload, b_k^{wm} è il valore del k-esimo bit del messaggio originale incorporato nel watermark, $\hat{b}_k \in \{0, 1\}$ è il bit estratto dal detector sull'immagine in esame, e $\mathbf{1}[\cdot]$ è la funzione indicatrice che vale 1 se i due bit coincidono e 0 altrimenti. Un attacco riuscito porta questa metrica verso 0.5: nell'ipotesi che i bit siano equiprobabili, tale valore corrisponde alla prestazione attesa di un estrattore che opera in assenza di qualsiasi segnale informativo, equivalente a un classificatore casuale.

- **Logit di presenza del watermark**: il valore `preds`[:, 0] restituito dall'extractor, ossia il logit associato alla classe *watermark present*. Un valore positivo indica che il detector classifica l'immagine come marcata; un valore negativo indica assenza della firma. Questa metrica è complementare alla bit accuracy: un attacco può abbassare la bit accuracy senza necessariamente ridurre il logit di presenza, o viceversa.

2.4.2 Architettura del sistema di watermarking PixelSeal

A differenza di SynthID, che pone la propria firma nel dominio della frequenza mediante una chiave di fase fissa per modello (Sezione 2.3.1), il sistema *VideoSeal* [25] sviluppato da Meta adotta un paradigma di tipo *encoder–decoder neurale*: il watermark non è definito esplicitamente nel dominio delle frequenze, bensì è una perturbazione additiva appresa nel dominio dei pixel, la cui struttura è distribuita e implicita nei pesi della rete. Per questo motivo gli approcci basati esclusivamente sul dominio frequenziale risultano meno efficaci, e la rimozione richiede a sua volta un attacco di tipo appreso, oggetto di questa sezione.

Il modello adottato per la sperimentazione è *PixelSeal* [15], appartenente alla famiglia VideoSeal e caratterizzato da un payload di 256 bit. Secondo quanto riportato dagli autori, PixelSeal stabilisce un nuovo stato dell'arte nel watermarking

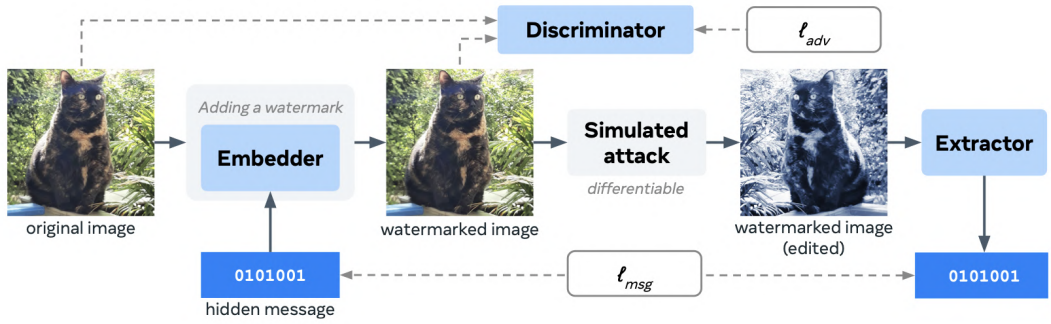
invisibile di immagini e video, mostrando miglioramenti sia in termini di robustezza sia di impercettibilità rispetto ai metodi precedenti [15].

A differenza del meccanismo a due blocchi analitici di SynthID, la pipeline VideoSeal è composta da due reti neurali addestrate **congiuntamente** in modalità *adversarial*, con uno stadio di trasformazioni applicato tra di esse per indurre robustezza (Figura 2.6).

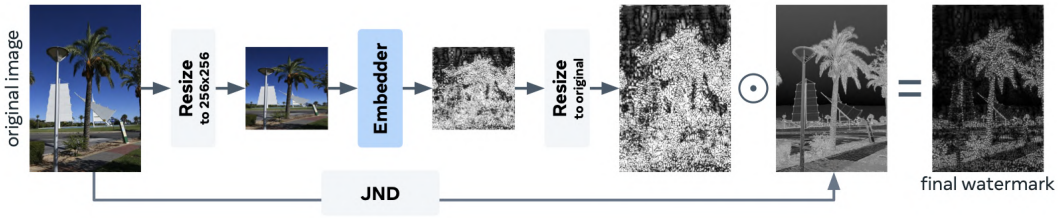
- **Embedder**: una rete U-Net che opera sulla componente di luminanza nello spazio *YUV*, uno spazio colore che separa la luminanza Y, ossia l'informazione di intensità, dalle due componenti di crominanza U e V, che codificano invece l'informazione cromatica. Un modulo di embedding del messaggio proietta i bit del payload in una rappresentazione latente che condiziona la generazione di un *residuo di watermark*. Tale residuo viene sommato all'immagine ospite da un modulo `blender` mediante un fattore di intensità `scaling_w`: il valore di default è 0.2, con valori tipici nell'intervallo $[0.1,\ 0.5]$; valori più elevati aumentano la robustezza del watermark a scapito dell'impercettibilità [25].

- **Attenuazione percettiva (JND)**: il modello *Just Noticeable Difference* produce una mappa di sensibilità visiva che assegna valori elevati alle regioni ad alta frequenza spaziale, quali contorni e texture, e valori bassi alle superfici uniformi. Moltiplicando il residuo di watermark per questa mappa, l'energia del watermark viene concentrata nelle regioni dove le variazioni di intensità sono meno percepibili dall'occhio umano, preservando le aree lisce e massimizzando l'impercettibilità complessiva.

- **Extractor**: una rete convoluzionale di tipo *ConvNeXt* [26] che, data un'immagine, estrae il messaggio incorporato. L'output è un tensore `preds` di forma $[B,\ 1 + K]$, dove B è la dimensione del batch e K è il numero di bit del payload: l'indice 0 contiene il logit associato alla classe *watermark present*, mentre gli indici $1{:}K$ contengono i logit dei singoli bit del payload, interpretati tramite soglia binaria a zero ($> 0 \Rightarrow$ bit 1, $\leq 0 \Rightarrow$ bit 0).



(a) Schema di addestramento *adversarial* di PixelSeal.



(b) Pipeline di embedding con attenuazione percettiva JND.

Figura 2.6: Pipeline di watermarking PixelSeal [15].

PixelSeal identifica e risolve tre limiti delle pipeline di watermarking neurale precedenti [15]:

1. **Proxy perceptual loss inaffidabili**: la dipendenza da metriche pixel-wise come l'MSE, che rimangono approssimazioni imperfette della percezione umana, ad esempio l'MSE penalizza allo stesso modo gli errori nelle aree lisce e in quelle testurizzate, mentre l'occhio umano è più sensibile agli artefatti nelle prime [15], introduce artefatti visibili del watermark. PixelSeal adotta un addestramento *adversarial-only*: l'impercettibilità non viene più imposta da loss esplicite calcolate pixel-per-pixel, bensì perseguita per via *adversarial* tramite il discriminatore.

2. **Instabilità di ottimizzazione**: gli obiettivi contrastanti di robustezza e qualità percettiva rendono il training instabile e dipendente da un tuning esaustivo degli iperparametri. PixelSeal introduce uno *schedule* a tre stadi che disaccoppia i due obiettivi, stabilizzando la convergenza.

3. **Gap di risoluzione**: robustezza e impercettibilità degradano scalando a immagini e video ad alta risoluzione. Il problema è affrontato tramite adattamento ad alta risoluzione, basato su attenuazione JND e su una tecnica che riproduce durante il training la stessa catena di operazioni applicata in fase di inferenza, eliminando lo scarto di distribuzione tra i due contesti e gli artefatti di sovracampionamento che ne derivano [15].

Per il video, PixelSeal estende l'approccio mediante *temporal watermark pooling*, evitando di marcare ogni singolo frame ad alta risoluzione.

2.4.3 Attacco a PixelSeal mediante denoising neurale

Dataset di addestramento

Per gli esperimenti di rimozione descritti in questa sezione è stata impiegata la sola categoria `real` del dataset (Sezione 2.2), ossia le 100 immagini reali prive di watermark, che fungono da riferimento pulito. A partire da queste si è costruito l'insieme di coppie (immagine con watermark, immagine pulita) necessario alla supervisione, applicando direttamente il modello PixelSeal come generatore del watermark. La procedura opera per ogni immagine sorgente come segue:

1. **Suddivisione in split**: le 100 immagini base sono ripartite in 50 per il *training*, 25 per la *validation* e 25 per il *test*.

2. **Data Augmentation**: da ciascuna immagine si estraggono quattro ritagli indipendenti. Ogni ritaglio è ottenuto in due fasi: un *random crop* alla dimensione del lato minimo comune al dataset, seguito da un *center crop* alla risoluzione finale di 256×256 pixel, corrispondente alla risoluzione di elaborazione nativa del modello PixelSeal. La patch così ottenuta costituisce l'immagine **pulita** di riferimento.

3. **Applicazione del watermark**: a ciascuna patch pulita viene applicato il watermark PixelSeal tramite la funzione `embed`, ottenendo la corrispondente versione **con watermark**.

Il ritaglio in quattro patch quadruplica la numerosità dei campioni, portando l'insieme finale a 200 coppie di training, 100 di validation e 100 di test. Ciascuna coppia $(\mathbf{x}_{wm}, \mathbf{x}_c)$ è composta dall'immagine con watermark $\mathbf{x}_{wm}$ e dalla corrispondente versione pulita $\mathbf{x}_c$ a parità di contenuto visivo, in modo che la loss di fedeltà,

$$\mathcal{L}_{\text{img}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{x}_i - x_{c,i}\right)^2,$$

dove N è il numero di pixel dell'immagine, $\hat{x}_i$ è il valore del pixel i-esimo prodotto dalla rete e $x_{c,i}$ è il valore del corrispondente pixel nell'immagine pulita di riferimento, sia definita su un riferimento semanticamente coerente con l'ingresso della rete. Il Listato 2.1 riporta il ciclo di generazione delle coppie.

Listato 2.1: Ritaglio in quattro patch e applicazione del watermark PixelSeal

```
for i in range(4):
    # Stage 1: random crop to the minimum side of the dataset
    top, left, h, w = T.RandomCrop.get_params(
        img, output_size=(min_size, min_size))
    random_cropped = TF.crop(img, top, left, h, w)

    # Stage 2: center crop to PixelSeal native resolution (256x256)
    final_cropped = TF.center_crop(
        random_cropped, output_size=(TARGET_SIZE, TARGET_SIZE))
    final_cropped.save(clean_save_path)  # clean reference patch

    # Apply PixelSeal watermark
    img_tensor = TF.to_tensor(final_cropped).unsqueeze(0).to(DEVICE)
    with torch.no_grad():
        embed_result = pixelseal.embed(img_tensor)
    wm_tensor = embed_result["imgs_w"].squeeze(0)  # already clamped to [0,1]
    TF.to_pil_image(wm_tensor.cpu()).save(wm_save_path)
```

Architettura della rete di attacco

L'attacco è realizzato da una rete encoder–decoder simmetrica (`UNetDenoiseAttack`) addestrata a ricostruire una versione *pulita* dell'immagine con watermark, distruggendo contestualmente il payload. La rete riceve in ingresso l'immagine marcata $\mathbf{x}_{wm}$ e produce l'immagine attaccata $\hat{\mathbf{x}}$.

- **Encoder**: blocco iniziale `DoubleConv` ($3 \to 16$ canali) seguito da sei stadi di downsampling (`MaxPool2d` + `DoubleConv`), con raddoppio progressivo dei

canali fino al *bottleneck* (512 → 1024). Ogni `DoubleConv` è la sequenza ($\text{Conv}_{3\times3}$ → BatchNorm → ReLU) × 2. (Listato 2.2)

Listato 2.2: Implementazione DoubleConv e stadi di Encoder

```
class DoubleConv(nn.Module):
    def __init__(self, in_channels, out_channels):
        super().__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1,
                bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1,
                bias=False),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True)
        )

self.inc   = DoubleConv(in_channels, 16)
self.down1 = nn.Sequential(nn.MaxPool2d(2), DoubleConv(16,  32))
self.down2 = nn.Sequential(nn.MaxPool2d(2), DoubleConv(32,  64))
self.down3 = nn.Sequential(nn.MaxPool2d(2), DoubleConv(64,  128))
self.down4 = nn.Sequential(nn.MaxPool2d(2), DoubleConv(128, 256))
self.down5 = nn.Sequential(nn.MaxPool2d(2), DoubleConv(256, 512))
self.bottleneck = nn.Sequential(nn.MaxPool2d(2), DoubleConv(512, 1024))
```

- **Decoder**: sei stadi di upsampling (`ConvTranspose2d`) con *skip connection* che concatenano le feature dell'encoder a pari risoluzione, reintegrando l'informazione spaziale fine persa durante il downsampling. (Listato 2.3)

Listato 2.3: Decoder con skip connection: definizione e forward

```
# Definition of upsampling stages
self.up1      = nn.ConvTranspose2d(1024, 512, kernel_size=2, stride=2)
self.conv_up1 = DoubleConv(1024, 512)  # 1024 = 512 (up) + 512 (skip x6)
self.up2      = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
self.conv_up2 = DoubleConv(512, 256)   # 512  = 256 (up) + 256 (skip x5)
self.up3      = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
self.conv_up3 = DoubleConv(256, 128)   # 256  = 128 (up) + 128 (skip x4)
self.up4      = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
self.conv_up4 = DoubleConv(128, 64)    # 128  =  64 (up) +  64 (skip x3)
self.up5      = nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2)
self.conv_up5 = DoubleConv(64, 32)     #  64  =  32 (up) +  32 (skip x2)
self.up6      = nn.ConvTranspose2d(32, 16, kernel_size=2, stride=2)
self.conv_up6 = DoubleConv(32, 16)     #  32  =  16 (up) +  16 (skip x1)

# Forward: x1...x6 are the encoder feature maps at each level,
# b is the bottleneck output.
# Each stage: upsampling + concatenation with the skip connection +
    DoubleConv
x1 = self.inc(x)            # 3   -> 16 channels
x2 = self.down1(x1)         # 16  -> 32
x3 = self.down2(x2)         # 32  -> 64
x4 = self.down3(x3)         # 64  -> 128
x5 = self.down4(x4)         # 128 -> 256
x6 = self.down5(x5)         # 256 -> 512
b  = self.bottleneck(x6)    # 512 -> 1024

t1 = self.conv_up1(torch.cat([self.up1(b),  x6], dim=1))
t2 = self.conv_up2(torch.cat([self.up2(t1), x5], dim=1))
t3 = self.conv_up3(torch.cat([self.up3(t2), x4], dim=1))
t4 = self.conv_up4(torch.cat([self.up4(t3), x3], dim=1))
t5 = self.conv_up5(torch.cat([self.up5(t4), x2], dim=1))
t6 = self.conv_up6(torch.cat([self.up6(t5), x1], dim=1))
```

- **Blocco di output**: convoluzione 1×1 ($16 \to 3$) seguita dall'attivazione $\sigma_{\text{out}}(z) = (\tanh(z) + 1)/2$, che vincola l'output nell'intervallo $[0, 1]$. (Listato 2.4)

Listato 2.4: Blocco di output con TanhNorm

```
class TanhNorm(nn.Module):
    def forward(self, x):
        return (torch.tanh(x) + 1) / 2

self.outc      = nn.Conv2d(16, out_channels, kernel_size=1)
self.activ_out = TanhNorm()
reconstructed_imgs = self.activ_out(self.outc(t6)) # Forward
```

Il detector PixelSeal è collegato a valle della rete, con i pesi congelati, in modo da estrarre i logit dei bit dell'immagine ricostruita e renderli disponibili al termine *adversarial* della funzione di costo (Figura 2.7).

Loss function e procedura di addestramento

La rete di attacco è stata addestrata secondo due configurazioni distinte, allo scopo di isolare il contributo del detector al processo di rimozione. Le due configurazioni si distinguono per la loss function e per la modalità di inizializzazione dei pesi; condividono invece tutti gli altri iperparametri. Sia $\hat{\mathbf{x}}$ l'immagine ricostruita dalla rete e $\mathbf{x}_c$ l'immagine pulita di riferimento.

Esperimento 1: U-Net senza detector. La rete è inizializzata casualmente e addestrata con la sola loss di fedeltà pixel-wise calcolata come errore assoluto medio (*Mean Absolute Error*, L1):

$$\mathcal{L}^{(1)} = \mathcal{L}^{\text{L1}}_{\text{img}} = \frac{1}{N}\sum_{i=1}^{N} \left|\hat{x}_i - x_{c,i}\right|,$$

dove N è il numero di pixel, $\hat{x}_i$ è il valore del pixel i-esimo prodotto dalla rete e $x_{c,i}$ è il valore del corrispondente pixel nell'immagine pulita di riferimento. Il detector non interviene nel processo di ottimizzazione. La U-Net si comporta dunque come un puro *denoising autoencoder*, e questo esperimento costituisce la *baseline* rispetto a cui misurare l'apporto del termine *adversarial*. L'ottimizzatore è Adam con learning rate fisso 2×10^{-5}, dimensione del batch 32 e gradient clipping con norma massima 1.0. L'addestramento era configurato per un massimo di 1000 epoche, con *early stopping* a patience 19 e *ReduceLROnPlateau* con fattore 0,5 e patience 10. Il checkpoint viene salvato ogni volta che la loss di validation raggiunge un nuovo minimo; l'addestramento si interrompe anticipatamente qualora non si osservino miglioramenti per 19 epoche consecutive. In questo esperimento l'early stopping è intervenuto all'epoca 301, a indicare che la loss di validation aveva smesso di migliorare e la rete aveva raggiunto la propria configurazione ottimale entro quel numero di epoche.

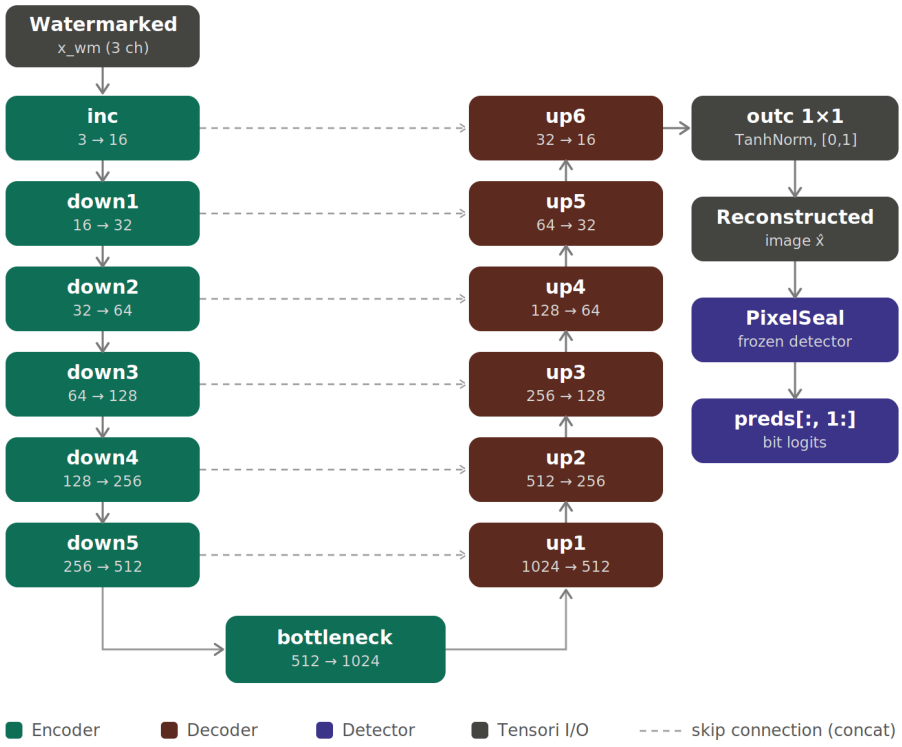


Figura 2.7: Architettura dell'attacco: U-Net di denoising a sei livelli con skip connection; il detector PixelSeal congelato fornisce i logit dei bit per la loss *adversarial*.

Esperimento 2: U-Net con detector. (Figura 2.7) La rete non è inizializzata casualmente, bensì a partire dal checkpoint prodotto dall'Esperimento 1: la U-Net parte quindi già da una ricostruzione fedele, e l'addestramento si concentra sull'effetto aggiuntivo introdotto dal termine *adversarial*. Il termine di fedeltà è ora calcolato come errore quadratico medio (*Mean Squared Error*, MSE):

$$\mathcal{L}_{\text{img}}^{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{x}_i - x_{c,i} \right)^2,$$

scelta motivata dalla maggiore sensibilità dell'MSE agli errori grandi, che si presta meglio all'ottimizzazione congiunta con il termine *adversarial* rispetto alla L1. A questo si affianca una loss *adversarial* che sfrutta il detector PixelSeal congelato.

Detto $\mathbf{z} = \text{extractor}(\hat{\mathbf{x}})$ il vettore dei logit dei bit estratti dall'immagine ricostruita, si penalizza lo scostamento della probabilità media dei bit dal valore di perfetta indecisione 0,5, che un'immagine con watermark distrutto dovrebbe esibire:

$$\mathcal{L}_{\text{adv}} = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{K}\sum_{k=1}^{K}\left|\,\sigma(z_{b,k}) - 0{,}5\,\right|,$$

dove B è la dimensione del batch, $K = 256$ il numero di bit del payload e σ la funzione sigmoide. La loss totale è:

$$\mathcal{L}^{(2)} = \mathcal{L}_{\text{img}}^{\text{MSE}} + w_{\text{adv}}\,\mathcal{L}_{\text{adv}},$$

con w_{adv} soggetto a un *warmup* lineare da 0 a 1 nelle prime 200 epoche, mantenuto a 1 per le successive 200:

$$w_{\text{adv}}(e) = \begin{cases} \dfrac{e}{200} & 1 \le e \le 200, \\ 1 & 200 < e \le 400. \end{cases}$$

Il warmup è motivato dalla necessità di stabilizzare l'ottimizzazione: nelle prime epoche il segnale *adversarial* è ancora mal calibrato, e introdurlo a piena intensità fin dall'inizio rischierebbe di degradare la fedeltà ricostruttiva già acquisita nell'Esperimento 1. Il gradiente della loss *adversarial* attraversa il detector congelato e si propaga fino alla sola U-Net. Tutti gli altri iperparametri (ottimizzatore, learning rate, batch size, gradient clipping e criteri di salvataggio del checkpoint) sono identici all'Esperimento 1, con la differenza che l'early stopping e lo scheduler adattivo del learning rate sono stati disabilitati: l'addestramento percorre l'intero arco di 400 epoche senza interruzioni anticipate, così da consentire l'osservazione completa dell'effetto del warmup *adversarial* sull'intera curva di loss.

Dettagli implementativi

I Listati 2.5 e 2.6 riportano rispettivamente l'inizializzazione dei modelli e il passo di addestramento su un singolo batch.

Listato 2.5: Inizializzazione dei modelli e dell'ottimizzatore

```
detector = videoseal.load("pixelseal").to(device)
detector.eval()
model = UNetDenoiseAttack(in_channels=3, out_channels=3, detector=detector).to(
    device)

# Load pre-trained checkpoint
ckpt = torch.load("checkpoints/unet_best_v4_0.pth", map_location=device,
    weights_only=True)
model.load_state_dict(ckpt)

# Freeze detector parameters
for param in detector.parameters():
    param.requires_grad = False

optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)
```

Listato 2.6: Passo di addestramento su un batch

```
# Forward pass
reconstructed_imgs, logits_reconstructed = model(wm_imgs)

# Fidelity loss: L1 in Experiment 1, MSE in Experiment 2
loss_img = criterion_img(reconstructed_imgs, clean_imgs)

# Adversarial loss (entropic bit balance) -- Experiment 2 only
probabilities     = torch.sigmoid(logits_reconstructed)
batch_bit_balance = probabilities.mean(dim=1)
loss_adv          = torch.abs(batch_bit_balance - 0.5).mean()

# Combined loss: w_img = 1.0 fixed, w_adv in [0,1] per warmup schedule
total_loss = 1.0 * loss_img + w_adv * loss_adv

# Backpropagation with gradient clipping (max_norm = 1.0)
total_loss.backward()
nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
optimizer.step()
```

2.4.4 Robustezza di PixelSeal agli attacchi tradizionali

L'attacco descritto nelle sezioni precedenti rappresenta solo una delle forme con cui l'integrità di un watermark può essere compromessa. Questa sezione caratterizza la robustezza di PixelSeal nei confronti di una batteria di *attacchi tradizionali*, ossia trasformazioni di elaborazione del segnale i cui parametri non sono appresi tramite ottimizzazione, ma fissati a priori o scelti da un insieme discreto di va-

lori predefiniti. Tali trasformazioni riproducono sia le manipolazioni accidentali tipiche della condivisione e dell'archiviazione delle immagini, quali compressione, ridimensionamento, conversioni di formato, sia tentativi deliberati di soppressione del watermark.

A differenza dell'attacco U-Net, che ottimizza i propri pesi contro il detector, gli attacchi tradizionali applicano una trasformazione la cui forma funzionale è deterministica e priva di parametri appresi, senza alcuna conoscenza del meccanismo di embedding. Il loro studio quantifica la soglia di degradazione che PixelSeal tollera prima del fallimento della decodifica, e fornisce il termine di confronto rispetto a cui valutare l'efficacia dell'attacco neurale.

Protocollo sperimentale

La procedura si articola in tre fasi:

1. **Watermarking**: a ciascuna immagine pulita viene applicato il watermark PixelSeal, producendo la corrispondente immagine con watermark.

2. **Baseline**: si estrae il messaggio dall'immagine pulita e da quella con watermark, registrando i rispettivi valori di *bit accuracy* e di logit di presenza. L'immagine con watermark intatta fornisce il riferimento superiore, corrispondente alla bit accuracy attesa in assenza di attacco, mentre l'immagine pulita fornisce il riferimento inferiore, corrispondente alla risposta del detector in assenza di qualsiasi segnale incorporato.

3. **Attacchi**: ogni immagine con watermark viene sottoposta a tutti gli attacchi della batteria; sull'immagine risultante si ri-estrae il messaggio e si misurano la bit accuracy residua e il logit di presenza, confrontandoli con la baseline. (Listato 2.7)

Listato 2.7: Applicazione di un attacco e misura della bit accuracy residua

```
def compute_ba(preds_dict, msg):
    preds = preds_dict["preds"]
    return ((preds[:, 1:] > 0).float() == msg).sum().item() / msg.numel()

att_pil, att_t = atk_fn(wm_t)
with torch.no_grad():
    det_a = model.detect(att_t)
ba_a    = compute_ba(det_a, msg)
logit_a = det_a["preds"][:, 0].item()
```

Tassonomia degli attacchi

La batteria comprende 47 attacchi, raggruppati per famiglia in base al dominio su cui agiscono (Tabella 2.3). Di seguito si descrivono le famiglie e i termini tecnici alla loro prima occorrenza.

- **Compressione lossy**: codifiche con perdita JPEG e WebP a fattori di qualità decrescenti. La quantizzazione dei coefficienti in frequenza scarta le componenti di dettaglio, su cui può annidarsi parte del residuo di watermark.

- **Rumore additivo**: *AWGN* (Additive White Gaussian Noise), ossia l'aggiunta a ogni pixel di rumore gaussiano a media nulla e deviazione standard crescente, e *salt and pepper noise*, consistente nella sostituzione casuale di una frazione di pixel con i valori estremi di nero e bianco.

- **Sfocatura e filtraggio spaziale**: sfocatura gaussiana e di *defocus*, *motion blur*, ottenuta convolvendo l'immagine con un kernel lineare orizzontale o diagonale, *sharpening* tramite *unsharp masking*, filtro *mediano*, che sostituisce ogni pixel con la mediana del suo intorno, e filtro *bilaterale*, che leviga preservando i contorni pesando i vicini per somiglianza di intensità.

- **Ritaglio e riscalatura**: ritaglio centrale a frazioni decrescenti dell'immagine seguito da reingrandimento alla risoluzione originale, e riduzione-reingrandimento a scala ridotta. Entrambi alterano l'allineamento spaziale su cui poggia la decodifica.

- **Resampling per interpolazione**: ingrandimento e successiva riduzione dell'immagine con quattro metodi di interpolazione: nearest neighbour, bilineare, bicubico, Lanczos. Introducono artefatti di ricampionamento di severità diversa.

- **Manipolazioni fotometriche e cromatiche**: *color jitter*, ossia perturbazione casuale di luminosità, contrasto e saturazione; correzione *gamma*, ovvero rimappatura non lineare delle intensità; *equalizzazione dell'istogramma*, cioè ridistribuzione dei livelli di luminanza per massimizzarne il contrasto; *stretching del contrasto*; e quantizzazione cromatica uniforme dei livelli RGB.

- **Manipolazione dei piani di bit (LSB)**: randomizzazione dei *Least Significant Bit*, i bit meno significativi di ciascun valore di pixel, che codificano le variazioni più fini e dove potrebbe risiedere parte del segnale di watermark.

- **Spazi colore alternativi**: iniezione di rumore sui canali di crominanza nello spazio *YCbCr* e quantizzazione nello spazio *HSV* (tonalità, saturazione, valore).

- **Trasformazioni geometriche**: rotazioni di piccola entità, che disallineano la griglia dei pixel.

- **Quantizzazione di palette**: riduzione dell'immagine a una tavolozza di colori limitata mediante l'algoritmo *median cut*, con e senza *dithering*, ovvero la dispersione controllata dell'errore di quantizzazione sui pixel adiacenti per simulare colori assenti dalla tavolozza.

Attacco sequenziale combinato

I 47 attacchi della tassonomia agiscono ciascuno su una singola dimensione del watermark e, applicati isolatamente, ne degradano la bit accuracy solo in misura limitata. L'analisi dei loro esiti individuali ha però permesso di identificare le trasformazioni più efficaci e di combinarle in un unico *attacco sequenziale*, che costituisce il 48-esimo e ultimo elemento della batteria. L'obiettivo è verificare l'effetto

Tabella 2.3: Famiglie di attacchi tradizionali applicati a PixelSeal.

Famiglia	**Varianti**	**N.**
Compressione lossy	JPEG (q75/50/30), WebP (q80/60/40)	6
Rumore additivo	AWGN (σ 0,05/0,10/0,20), sale e pepe	4
Sfocatura e filtraggio	gaussiana, defocus, motion (H/D), sharpening, mediano, bilaterale	8
Ritaglio e riscalatura	center crop (85/70/50%), scaling (0,75/0,50)	5
Resampling	nearest, bilineare, bicubico, Lanczos	4
Fotometrici e cromatici	color jitter, gamma (1,15/0,85), equalizzazione, contrast stretch, quant. RGB	7
Bit-plane (LSB)	randomizzazione di 2/3/4 bit	3
Spazi colore alternativi	rumore crominanza YCbCr, quantizzazione HSV	2
Geometrici	rotazione (2°/5°)	2
Quantizzazione di palette	median cut 256/128/64 colori, con/senza dithering	6
Totale		**47**

cumulativo della loro applicazione in cascata, sotto l'ipotesi che un watermark robusto a ciascuna trasformazione presa singolarmente possa non sopravvivere alla loro composizione.

La pipeline applica in sequenza le seguenti sette operazioni:

$$\texttt{GammaCorr. 0.85} \rightarrow \texttt{ColorJitter} \rightarrow \texttt{ChromaQuant32} \rightarrow \texttt{HSV\_Quant} \rightarrow \texttt{PNGQuant 64D} \rightarrow \texttt{JPEG q30} \rightarrow \texttt{WebP q40}$$

La scelta dell'ordine non è casuale, ma riflette una progressione tra domini di alterazione differenti, così da massimizzare la superficie d'attacco. Le prime due operazioni agiscono nel dominio *fotometrico*: la correzione gamma a 0,85 rimappa in modo non lineare le intensità, mentre il *color jitter* perturba luminosità, contrasto e saturazione. Le due successive operano nel dominio dello *spazio colore*: la quantizzazione cromatica a 32 livelli (ChromaQuant32) e la quantizzazione nello spazio HSV riducono la risoluzione cromatica su cui può poggiare parte del segnale. La quantizzazione di palette *median cut* a 64 colori con dithering (PNGQuant 64D) comprime ulteriormente la gamma cromatica disponibile. Le ultime due operazioni introducono infine artefatti di *compressione lossy* su due basi di trasformazione distinte: la quantizzazione DCT della compressione JPEG a qualità 30 e la diversa base di trasformazione della compressione WebP a qualità 40. L'applicazione in sequenza di due codec con basi di trasformazione distinte forza una doppia riquantizzazione spettrale, riducendo ulteriormente la coerenza del segnale residuo che avesse superato i passi precedenti.

Capitolo 3

Esperimenti e Risultati

Dopo aver descritto nel Capitolo 2 le tecnologie di watermarking impiegate e le tecniche di attacco, questo capitolo presenta gli esperimenti condotti e ne discute i risultati. La trattazione segue lo stesso ordine dei sistemi analizzati: si parte dalla rimozione di SynthID, per poi passare a PixelSeal.

3.1 Rimozione del watermark SynthID

L'attacco di rimozione del watermark SynthID è stato condotto mediante la pipeline *reverse-SynthID* V4 descritta nella Sezione 2.3.1, nella sua configurazione di intensità bilanciata.

3.1.1 Configurazione sperimentale

L'elaborazione è stata condotta applicando separatamente la pipeline di rimozione alle due categorie di immagini generate, **text-to-image** (T2I) e **image+text-to-image** (IT2I), ciascuna composta da 100 immagini. In entrambe le esecuzioni sono stati impiegati il codebook spettrale V4, il modello generatore **nano-banana-pro-preview** e il preset di intensità `final`, corrispondente alla configurazione bilanciata della pipeline a sette stadi. Va sottolineato che la valutazione di un attacco a SynthID presenta una difficoltà intrinseca rispetto al caso PixelSeal: in assenza di accesso al sistema di verifica ufficiale di Google, la verifica dell'esito della rimozione non può basarsi su una metrica numerica diretta come la bit accuracy,

ma richiede di interrogare il servizio di generazione NanoBanana per ciascuna immagine attaccata, rendendo la raccolta dei dati significativamente più onerosa e dipendente dalla disponibilità del servizio esterno. Al termine dell'elaborazione, ciascuna immagine è stata sottoposta alla verifica di provenienza di Gemini, che utilizza il sistema di verifica SynthID per stabilire se il watermark sia ancora rilevabile. Nel presente lavoro una rimozione è considerata riuscita quando il sistema di verifica SynthID non rileva più la presenza del watermark.

3.1.2 Risultati

La Figura 3.1 riporta l'esito della verifica per le due categorie. La pipeline, basata esclusivamente su elaborazione del segnale, ottiene una rimozione rilevabile solo in una minoranza dei casi, con una marcata asimmetria tra le categorie: il 29% di successi sugli IT2I contro appena l'1% sui T2I. In entrambi i casi il watermark SynthID si conferma in larga parte resistente a questo tipo di attacco, ma le immagini T2I risultano nettamente più resistenti di quelle IT2I, con il watermark che persiste nel 99% dei casi.

L'asimmetria osservata costituisce il dato di maggiore interesse: le immagini generate da solo testo si comportano come se incorporassero una firma più stabile rispetto a quelle ottenute condizionando il modello su un'immagine di partenza. Sono plausibili almeno tre interpretazioni, che richiederebbero tuttavia una verifica sperimentale dedicata.

La prima è che il processo di editing alla base delle IT2I alteri o indebolisca parzialmente la coerenza di fase del watermark, rendendolo più esposto alla sottrazione spettrale: il condizionamento sull'immagine di partenza potrebbe interferire con la capacità del generatore di imprimere il segnale in modo uniforme, riducendone la robustezza strutturale.

La seconda riguarda la risoluzione: le IT2I, ereditando risoluzione e proporzioni dall'immagine di condizionamento, presentano una maggiore eterogeneità rispetto alle T2I (Sezione 2.3.1). Poiché le frequenze portanti di SynthID dipendono dalla risoluzione, il codebook V4 può risultare meno calibrato per alcune IT2I, rendendo l'attacco spettrale più efficace su quella categoria. Nelle T2I invece la firma risul-

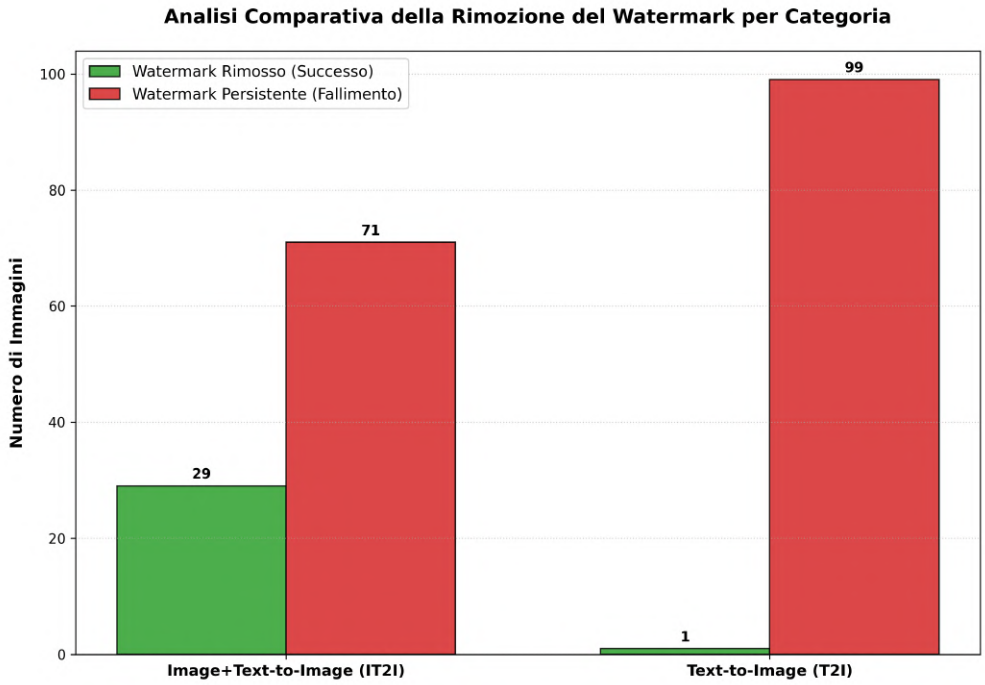


Figura 3.1: Analisi comparativa della rimozione del watermark SynthID per categoria di generazione. Per ciascuna delle due categorie, le barre indicano il numero di immagini in cui il watermark è stato rimosso con successo, ossia non più rilevato dalla verifica di Gemini (in verde), e quelle in cui il watermark è invece persistita all'attacco (in rosso). La rimozione riesce nel 29% degli IT2I e nell'1% dei T2I, evidenziando la maggiore robustezza del watermark nelle immagini generate da solo testo.

ta impressa sulla risoluzione nativa del generatore, con un profilo spettrale ben coperto dal codebook.

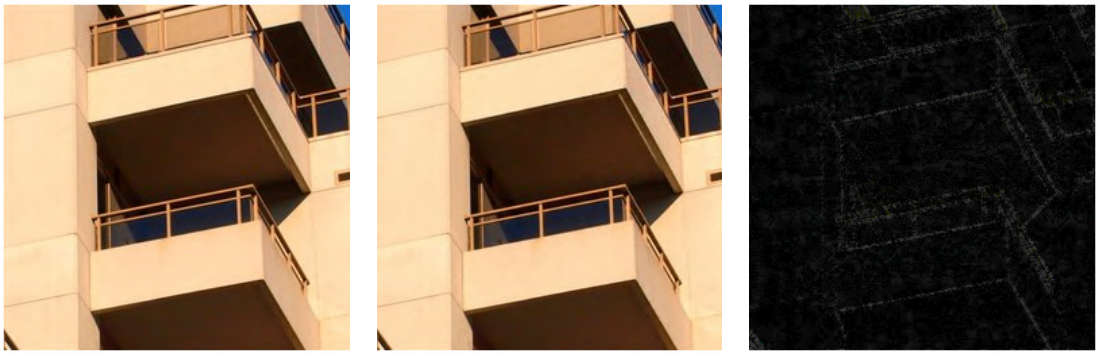
La terza ipotesi, più generale, è che le caratteristiche visive dell'immagine di condizionamento influenzino direttamente la qualità del watermark incorporato: immagini con texture complesse, dominanti cromatiche particolari o strutture ad alta frequenza potrebbero interferire con il meccanismo di embedding di SynthID, riducendo la coerenza del segnale indipendentemente dalla risoluzione.

Tale lettura rimane a livello di ipotesi: il campione è limitato a 100 immagini per categoria, è stato impiegato un singolo preset di intensità dell'attacco, e le immagini di condizionamento non sono state selezionate in modo controllato rispetto alle loro caratteristiche visive. Una caratterizzazione più solida richiederebbe di

variare sistematicamente questi tre fattori.

3.2 Attacco a PixelSeal

Prima di analizzare gli attacchi, è opportuno valutare l'impatto visivo del watermark incorporato sull'immagine. La Figura 3.2 mostra un'immagine del dataset nell'originale, nella versione watermarked e nella mappa di differenza amplificata di un fattore 10.



(a) Immagine originale. (b) Immagine con watermark PixelSeal. (c) Differenza assoluta amplificata $\times 10$.

Figura 3.2: Impercettibilità del watermark PixelSeal.

La mappa di differenza rivela un pattern distribuito sull'intera immagine, con energia concentrata nelle regioni ad alta frequenza spaziale, quali superfici texturizzate e zone di transizione cromatica: questo riflette la natura del residuo additivo appreso dal modello di embedding, che concentra il segnale nelle aree dove le variazioni di intensità sono meno percepibili dall'occhio umano, in accordo con la maschera JND descritta nella Sezione 2.4.2.

3.2.1 Attacchi tradizionali individuali

Oltre all'attacco basato su denoising neurale, è stata condotta una valutazione sistematica della robustezza di PixelSeal nei confronti di **47 attacchi tradizionali**,dai quali è stato successivamente derivato un attacco composito, per un totale di **48 configurazioni di attacco**, organizzati in dieci macrocategorie: compressione

lossy, rumore additivo, filtri di sfocatura, ritaglio e ridimensionamento, resampling per interpolazione, variazioni fotometriche, perturbazione dei bit meno significativi (LSB), trasformazioni in spazi colore alternativi, rotazioni geometriche e quantizzazione della palette cromatica mediante algoritmo median cut. Ogni attacco è stato applicato all'intero dataset di 100 immagini naturali ad alta risoluzione e già contenenti il watermark PixelSeal. La qualità della rilevazione è misurata dalla *Bit Accuracy* (BA), come descritto nella sezione 2.4.1. La soglia operativa adottata da Soucek e colleghi [15] per classificare il watermark come *ancora rilevabile* è BA > 0.70. Tale soglia è giustificata dagli autori in relazione alla capacità di correzione degli errori del sistema: al di sopra di essa il payload rimane recuperabile con affidabilità sufficiente, mentre al di sotto la decodifica del messaggio non è più garantita [15]. Per ciascun attacco sono riportati la media μ_{BA}, la deviazione standard σ_{BA} la percentuale di immagini classificate con watermark ancora rilevabile.

Panoramica dei risultati

La Figura 3.3 riassume graficamente, per ciascuno dei 48 attacchi, la percentuale di immagini che conservano il watermark dopo la trasformazione. Le barre sono raggruppate per macrocategoria e codificate cromaticamente.

Dall'analisi globale emergono i seguenti punti chiave:

- In **40 casi su 48** la percentuale di immagini che mantengono il watermark è pari al 100%.: ogni immagine del dataset supera la soglia senza eccezioni.
- Solo **3 attacchi individuali** riducono la percentuale di immagini che mantengono il watermark di oltre il 5%: AWGN σ=0.20 (25%), Motion Blur Diagonale (74%) e AWGN σ=0.10 (90%).
- L'attacco con la **deviazione standard più elevata** è il Motion Blur Diagonale ($\sigma_{\text{BA}} = 0.179$), suggerendo una maggiore variabilità dell'effetto dell'attacco tra immagini differenti.

La Tabella 3.1 riporta i valori numerici completi per tutti gli attacchi.

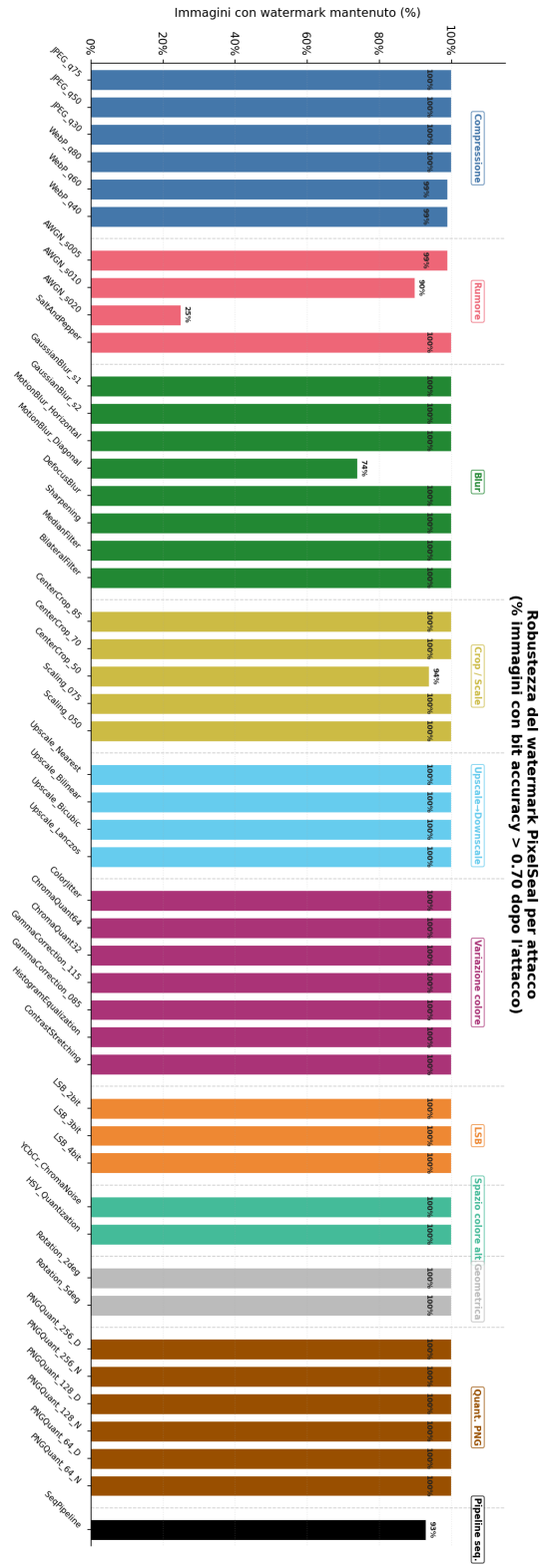


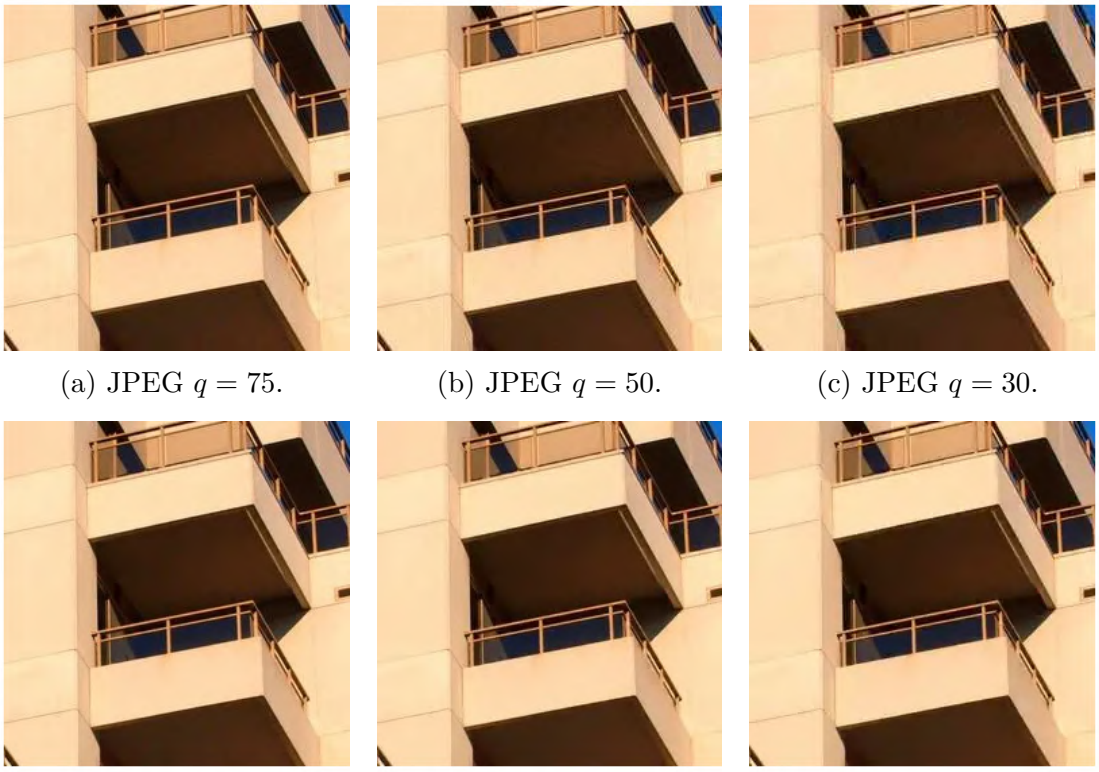
Figura 3.3: Robustezza di PixelSeal sotto 48 attacchi tradizionali. Ogni barra indica la percentuale di immagini (su 100) con BA $>$ 0.70 dopo l'attacco. Le barre sono raggruppate per macrocategoria con codifica cromatica consistente con le figure successive.

Tabella 3.1: Bit accuracy media sul dataset di test ($n = 100$ immagini), deviazione standard e percentuale di immagini con watermark ancora rilevabile (BA > 0.70) per ciascun attacco testato. La soglia 0.70 è quella adottata da Souček et al. [15].

Macrocategoria	**Attacco**	μ_{BA}	σ_{BA}	**% WM**
Compressione	JPEG q75	0.9973	0.0055	100
	JPEG q50	0.9904	0.0138	100
	JPEG q30	0.9646	0.0353	100
	WebP q80	0.9895	0.0277	100
	WebP q60	0.9715	0.0520	99
	WebP q40	0.9468	0.0574	99
Rumore	AWGN σ=0.05	0.9485	0.0540	99
	AWGN σ=0.10	0.8166	0.0923	90
	AWGN σ=0.20	0.6382	0.0963	25
	Salt & Pepper	0.9053	0.0500	100
Blur / Nitidezza	Gaussian Blur σ=1.5	0.9976	0.0062	100
	Gaussian Blur σ=3.0	0.9940	0.0111	100
	Motion Blur orizz.	0.9916	0.0170	100
	Motion Blur diag.	0.8097	0.1793	74
	Defocus Blur	0.9787	0.0306	100
	Sharpening	0.9966	0.0085	100
	Median Filter	0.9976	0.0059	100
	Bilateral Filter	0.9966	0.0089	100
Crop / Scale	Center Crop 85%	0.9950	0.0110	100
	Center Crop 70%	0.9850	0.0232	100
	Center Crop 50%	0.8714	0.0927	94
	Scaling 75%	0.9985	0.0040	100
	Scaling 50%	0.9984	0.0040	100
Resampling	Nearest Neighbor	0.9986	0.0037	100
	Bilineare	0.9982	0.0044	100
	Bicubico	0.9986	0.0038	100
	Lanczos	0.9986	0.0039	100
Variazione colore	Color Jitter	0.9978	0.0054	100
	Chroma Quant. 64	0.9978	0.0050	100
	Chroma Quant. 32	0.9937	0.0147	100
	Gamma γ=1.15	0.9985	0.0039	100
	Gamma γ=0.85	0.9981	0.0048	100
	Hist. Equalization	0.9958	0.0075	100
	Contrast Stretching	0.9980	0.0054	100
Randomizzazione LSB	LSB 2 bit	0.9978	0.0054	100
	LSB 3 bit	0.9933	0.0099	100
	LSB 4 bit	0.9652	0.0375	100
Spazio colore alt.	YCbCr Chroma Noise	0.9967	0.0056	100
	HSV Quantization	0.9856	0.0194	100
Geometrica	Rotazione 2°	0.9976	0.0062	100
	Rotazione 5°	0.9971	0.0067	100
Palette quantization (PNG, Median Cut)	PNG 256 col. (dither)	0.9957	0.0067	100
	PNG 256 col. (no dither)	0.9957	0.0067	100
	PNG 128 col. (dither)	0.9929	0.0099	100
	PNG 128 col. (no dither)	0.9929	0.0099	100
	PNG 64 col. (dither)	0.9838	0.0194	100
	PNG 64 col. (no dither)	0.9838	0.0194	100
Pipeline seq.	SeqPipeline (7 step)	0.8355	0.0965	93

Compressione lossy

La Figura 3.4 mostra l'effetto visivo della compressione JPEG e WebP a diversi livelli di qualità.



(a) JPEG $q = 75$. (b) JPEG $q = 50$. (c) JPEG $q = 30$.

(d) WebP $q = 80$. (e) WebP $q = 60$. (f) WebP $q = 40$.

Figura 3.4: Effetto della compressione lossy JPEG e WebP a qualità decrescente.

Gli algoritmi di compressione lossy non compromettono la rilevazione del watermark in nessuno dei sei casi testati. La BA media si mantiene superiore a 0.94 anche alla qualità minima (JPEG q30: $\mu_{\text{BA}} = 0.965$; WebP q40: $\mu_{\text{BA}} = 0.947$). I due casi con percentuale di immagini con watermark ancora rilevabile pari al 99% invece del 100% (WebP q60 e WebP q40) sono attribuibili alla diversa struttura degli artefatti introdotti dal codec WebP rispetto a JPEG: mentre JPEG opera una quantizzazione dei coefficienti DCT su blocchi 8×8, WebP si basa su VP8, un codec video che utilizza una decomposizione in blocchi di dimensione variabile (4×4 fino a 16×16) con predizione intra-frame, producendo artefatti spazialmente più irre-

golari che possono risultare localmente più distruttivi per specifiche componenti a bassa energia del segnale watermark. La **deviazione standard cresce al diminuire della qualità** (da 0.006 per JPEG q75 a 0.057 per WebP q40), riflettendo una maggiore variabilità del degrado tra immagini con contenuto diverso.

Rumore additivo gaussiano (AWGN)

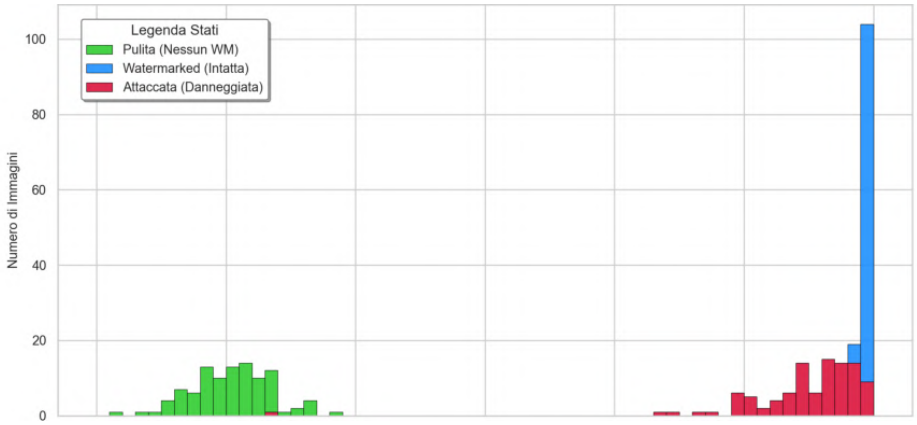
Il rumore gaussiano additivo è l'attacco più efficace nel ridurre la Bit Accuracy tra quelli valutati. La Figura 3.5 mostra la progressione del degrado visivo al crescere della deviazione standard, mentre la Figura 3.6 riporta le distribuzioni di BA per i tre livelli di rumore.

(a) Immagine con watermark (riferimento). (b) AWGN $\sigma = 0{,}05$. (c) AWGN $\sigma = 0{,}10$. (d) AWGN $\sigma = 0{,}20$.

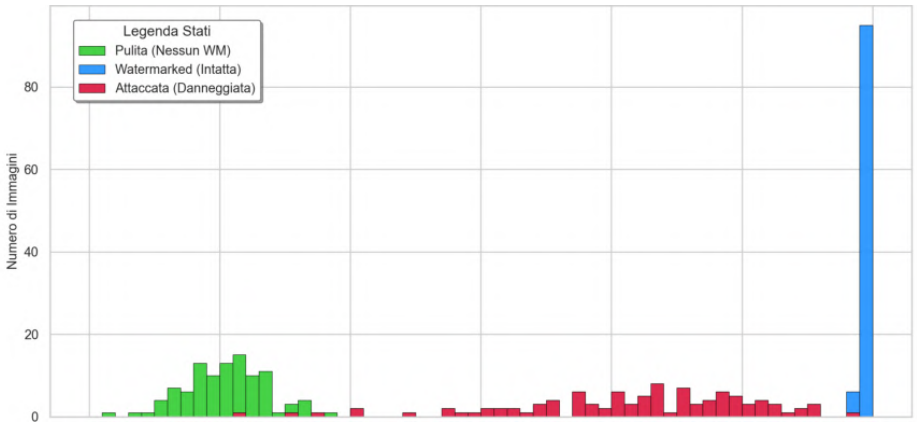
Figura 3.5: Effetto del rumore gaussiano additivo (AWGN) a intensità crescente.

Con σ=0.05 l'effetto è marginale: la distribuzione di BA delle immagini attaccate rimane concentrata vicino a 1.0 ($\mu_{\text{BA}} = 0.949$, % WM = 99%). Con σ=0.10 la distribuzione mostra una dispersione sensibilmente maggiore ($\mu_{\text{BA}} = 0.817$, $\sigma_{\text{BA}} = 0.092$) e il 10% delle immagini scende sotto soglia. Con σ=0.20, un livello di degrado visivo già evidente, la BA si riduce a $\mu_{\text{BA}} = 0.638$ e **solo il 25% delle immagini** mantiene il watermark correttamente decodificabile.

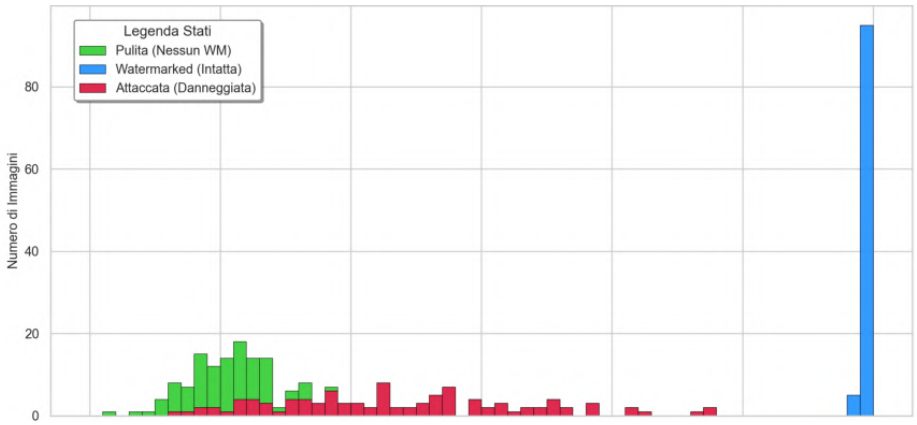
La Figura 3.7 illustra il caso peggiore e il caso migliore osservati con AWGN σ=0.20, evidenziando come la resistenza all'attacco dipenda dal contenuto dell'immagine.



(a) AWGN $\sigma = 0{,}05$.



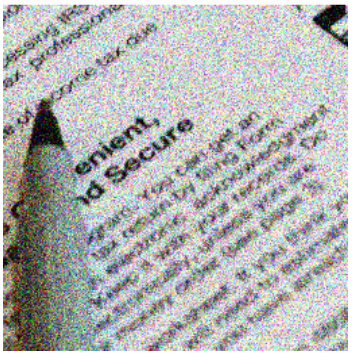
(b) AWGN $\sigma = 0{,}10$.



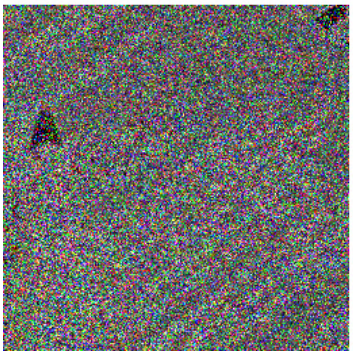
(c) AWGN $\sigma = 0{,}20$.

Figura 3.6: Distribuzioni di bit accuracy per i tre livelli di AWGN. Al crescere di σ la distribuzione delle immagini attaccate si sposta progressivamente verso il livello di puro caso (BA $\approx 0{,}5$).

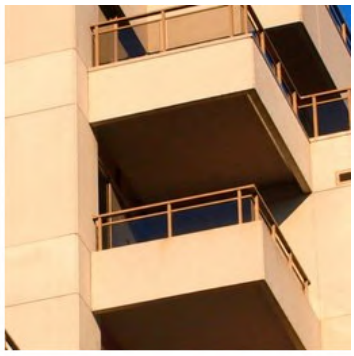
(a) Caso peggiore: immagine con watermark.



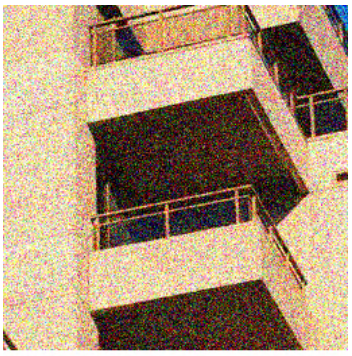
(b) Caso peggiore: immagine attaccata (BA = 0,476).



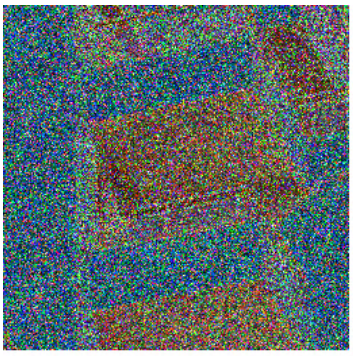
(c) Caso peggiore: differenza amplificata ×3.



(d) Caso migliore: immagine con watermark.



(e) Caso migliore: immagine attaccata (BA = 0,969).

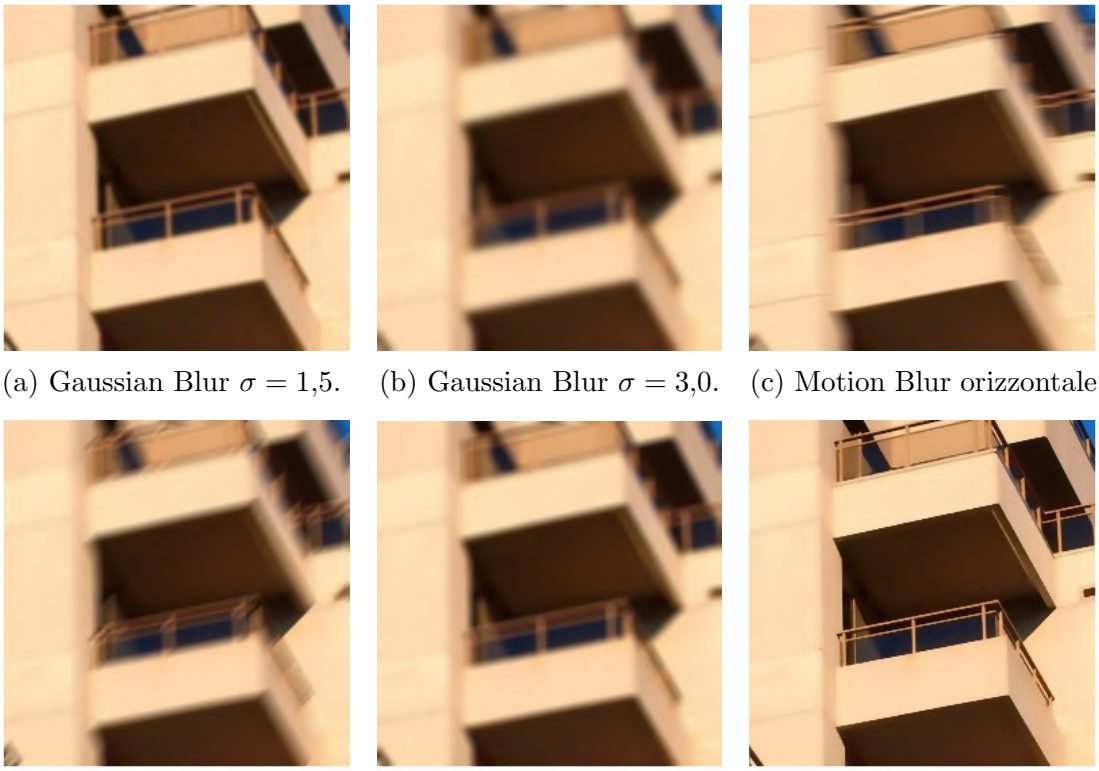


(f) Caso migliore: differenza amplificata ×3.

Figura 3.7: AWGN $\sigma = 0{,}20$: confronto tra il caso peggiore e il caso migliore osservati sul dataset.

La vulnerabilità all'AWGN intenso trova giustificazione nelle proprietà del segnale incorporato: il watermark è un segnale a bassa energia e il rumore ad alta ampiezza agisce come un'interferenza a banda larga sull'intero spettro spaziale, riducendo il rapporto segnale-rumore del messaggio incorporato al di sotto della soglia di decodifica.

Blur e filtraggio



(a) Gaussian Blur $\sigma = 1{,}5$. (b) Gaussian Blur $\sigma = 3{,}0$. (c) Motion Blur orizzontale.

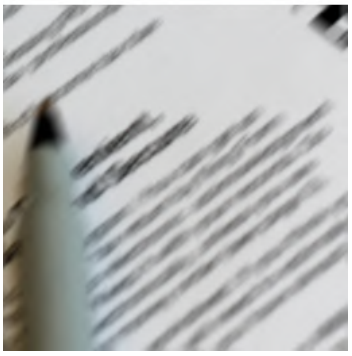
(d) Motion Blur diagonale. (e) Defocus Blur. (f) Bilateral Filter.

Figura 3.8: Effetto delle operazioni di sfocatura e filtraggio sul watermark PixelSeal.

Tra gli otto operatori di blur testati (Figura 3.8), solo il **Motion Blur Diagonale** produce una riduzione significativa della robustezza. Di particolare interesse è la deviazione standard $\sigma_{\text{BA}} = 0.179$: il valore massimo osservato negli attacchi effettuati. Questo valore segnala un comportamento fortemente eterogeneo: alcune immagini resistono perfettamente all'attacco, mentre altre perdono quasi completamente il watermark.

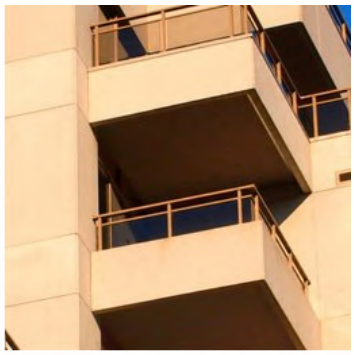
La Figura 3.9 mostra questa eterogeneità: il caso peggiore osservato raggiunge BA $= 0.445$ su un'immagine caratterizzata da una predominanza di strutture orientate orizzontalmente, che il kernel diagonale distorce maggiormente, mentre il caso migliore mantiene BA $= 1.00$.

(a) Caso peggiore: immagine con watermark.

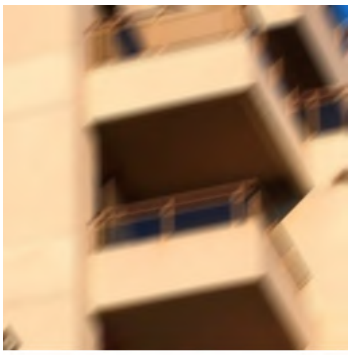


(b) Caso peggiore: immagine attaccata (BA = 0,445).

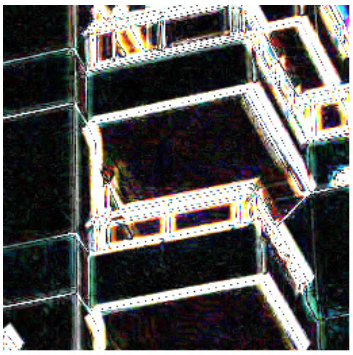
(c) Caso peggiore: differenza amplificata ×12.



(d) Caso migliore: immagine con watermark.



(e) Caso migliore: immagine attaccata (BA = 1,000).

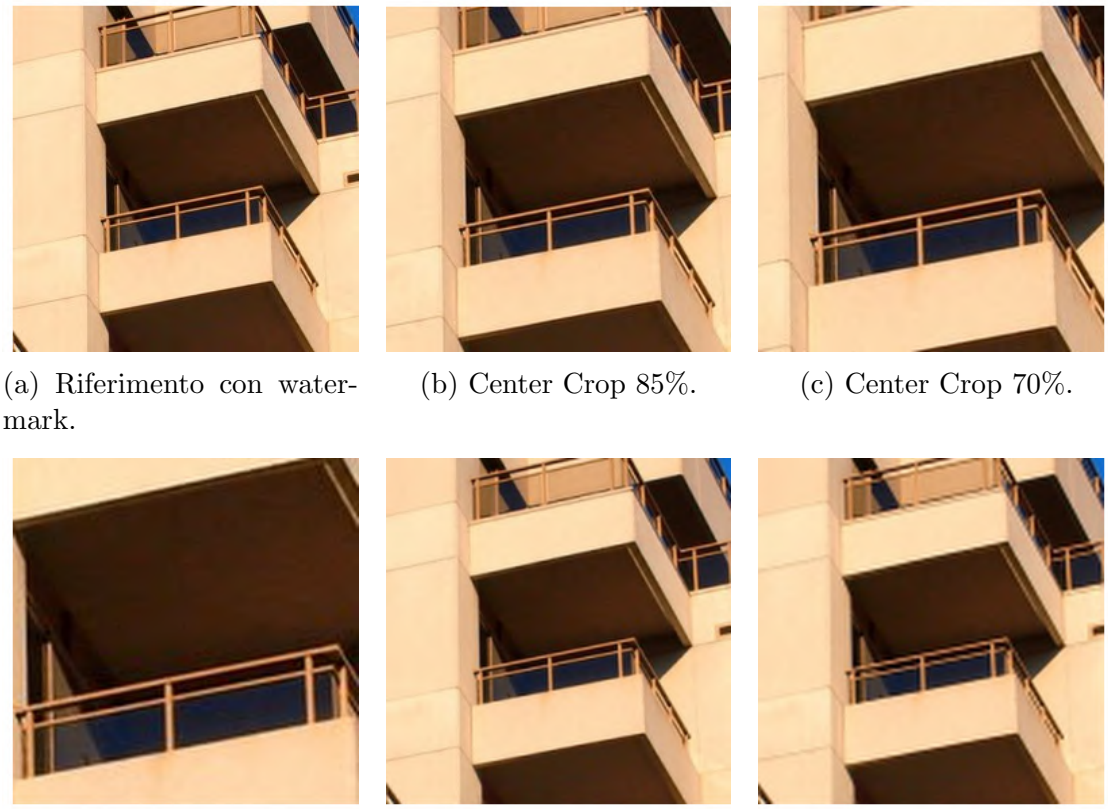


(f) Caso migliore: differenza amplificata ×12.

Figura 3.9: Motion Blur Diagonale: eterogeneità del comportamento tra immagini diverse.

Questo suggerisce che la suscettibilità al Motion Blur Diagonale sia correlata alla struttura spaziale locale: immagini con energia prevalente nelle frequenze orizzontali sono più vulnerabili perché il kernel 11×11 a 45° attenua maggiormente tali componenti. Al contrario, immagini ricche di bordi diagonali e verticali mostrano una degradazione sensibilmente inferiore all'attacco. Tutti gli altri operatori, inclusi il Defocus Blur con raggio 4.0 e il Bilateral Filter, non causano alcuna perdita di watermark.

Ritaglio e ridimensionamento



(a) Riferimento con watermark. (b) Center Crop 85%. (c) Center Crop 70%.

(d) Center Crop 50%. (e) Scaling 75%. (f) Scaling 50%.

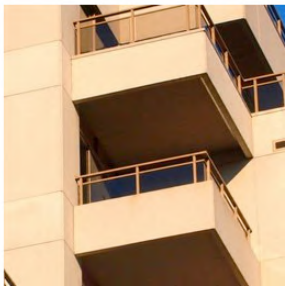
Figura 3.10: Effetto del ritaglio centrato e del ridimensionamento sul watermark PixelSeal.

Come mostrato in Figura 3.10, il Center Crop al 50% è la trasformazione più severa, con una percentuale di watermark residuo pari a 94% e $\mu_{\text{BA}} = 0.871$.

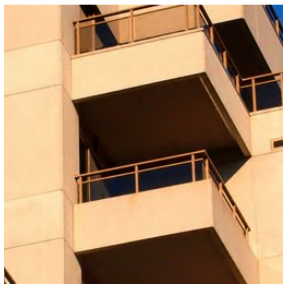
Considerando che questa trasformazione elimina il 75% dei pixel originali e introduce interpolazione LANCZOS per il resize, il risultato evidenzia una robustezza elevata.

Lo **Scaling al 50%** ($\mu_{\text{BA}} = 0.998$, $\%\,\text{WM} = 100\%$) risulta significativamente meno dannoso del Center Crop al 50%, pur riducendo la risoluzione a un quarto dei pixel complessivi. La differenza è strutturale: il downscaling isotropo preserva maggiormente la struttura globale del segnale watermark su tutta l'immagine ridotta, preservandone la coerenza globale; il ritaglio invece rimuove fisicamente porzioni di immagine, eliminando definitivamente la porzione di segnale contenuta nelle aree rimosse.

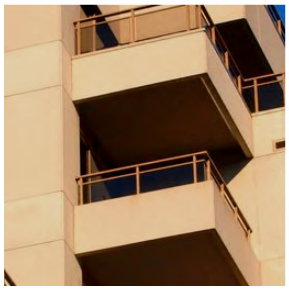
Variazioni fotometriche



(a) Gamma $\gamma = 0{,}85$.

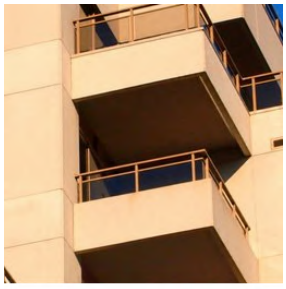


(b) Gamma $\gamma = 1{,}15$.

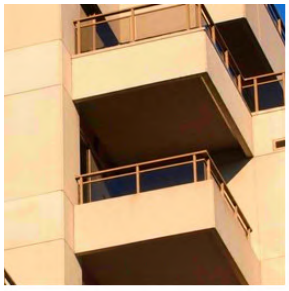


(c) Color Jitter.

(d) Histogram Equalization.



(e) Contrast Stretching.



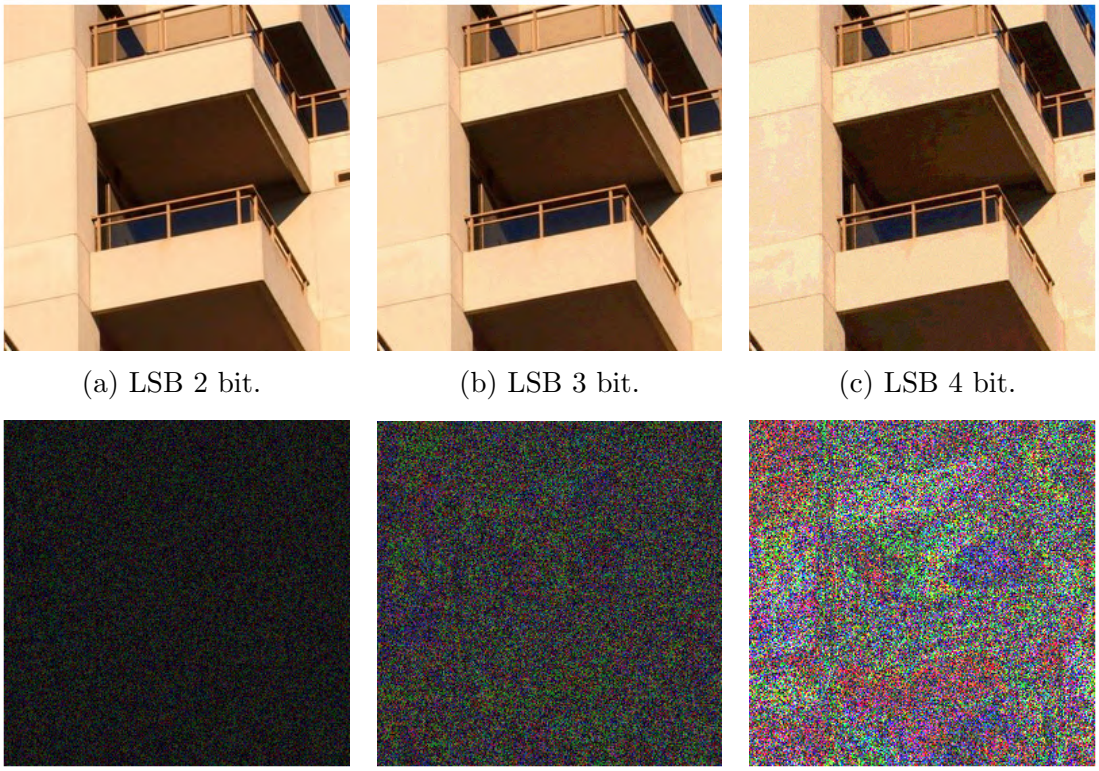
(f) Chroma Quantization 32 livelli.

Figura 3.11: Effetto delle variazioni fotometriche e cromatiche sul watermark PixelSeal.

La Figura 3.11 mostra che l'intera categoria delle variazioni fotometriche non produce alcun caso di fallimento della decodifica del watermark. Tutte le BA me-

die superano 0.985 con deviazioni standard molto contenute ($\sigma < 0.016$). Questo risultato suggerisce che il watermark sia distribuito spazialmente in modo sufficientemente ridondante da tollerare modifiche fotometriche globali come l'equalizzazione dell'istogramma ($\mu_{\text{BA}} = 0.996$), che modifica significativamente la distribuzione dei livelli di luminanza.

Randomizzazione LSB



(a) LSB 2 bit. (b) LSB 3 bit. (c) LSB 4 bit.

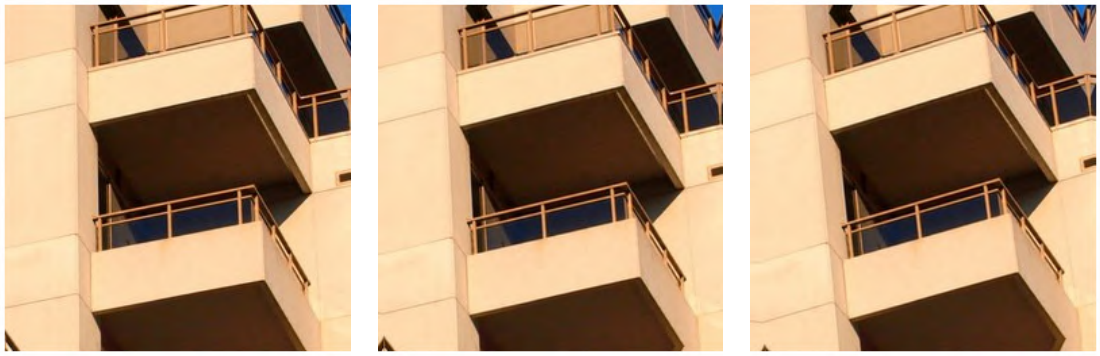
(d) Differenza ×20, LSB 2 bit. (e) Differenza ×20, LSB 3 bit. (f) Differenza ×20, LSB 4 bit.

Figura 3.12: Randomizzazione LSB a 2, 3 e 4 bit: immagini attaccate (riga superiore) e mappe di differenza ampliate ×20 rispetto all'immagine con watermark (riga inferiore).

La Figura 3.12 illustra come la randomizzazione dei bit meno significativi produca mappe di differenza con rumore crescente ma non comprometta il watermark nemmeno nel caso più aggressivo (LSB 4 bit: $\mu_{\text{BA}} = 0.965$, tutte le immagini con

watermark ancora rilevabile). PixelSeal non utilizza i bit meno significativi come sede esplicita dell'embedding, ma apprende un residuo distribuito sull'immagine tramite l'embedder.

Trasformazioni geometriche



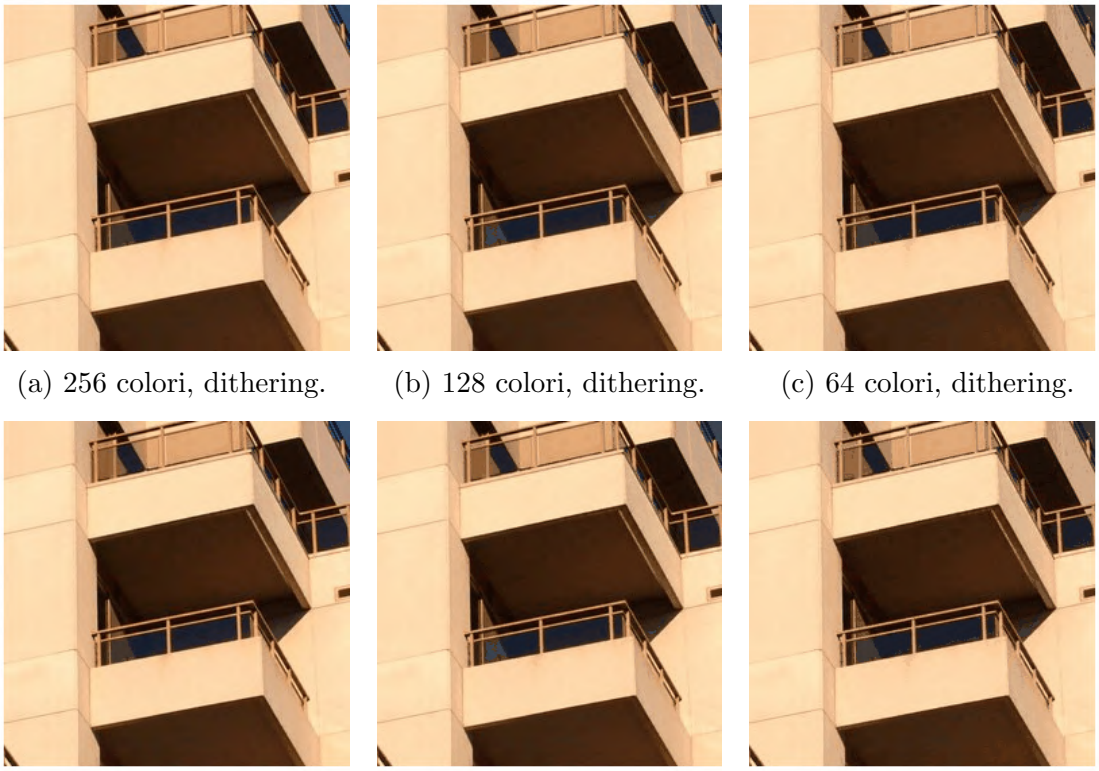
(a) Riferimento con watermark. (b) Rotazione 2. (c) Rotazione 5.

Figura 3.13: Effetto delle rotazioni geometriche sul watermark PixelSeal.

Le rotazioni di piccola entità non compromettono il watermark in nessuno dei due casi testati, come mostrato in Figura 3.13. Con una rotazione di 2 ($\mu_{\text{BA}} = 0.998$) e di 5 ($\mu_{\text{BA}} = 0.997$) il watermark rimane perfettamente rilevabile nel 100% delle immagini. Questo risultato è coerente con il meccanismo di embedding di PixelSeal: il residuo additivo è distribuito sull'intera immagine e non è ancorato a una griglia spaziale rigida, rendendolo insensibile ai piccoli disallineamenti geometrici introdotti dalla rotazione.

Quantizzazione adattiva della palette PNG

Un risultato degno di nota emerge dalla Figura 3.14: le varianti con dithering Floyd-Steinberg e senza dithering producono valori di BA **identici** a parità di numero di colori (ad esempio, PNGQuant 64: $\mu_{\text{BA}} = 0.984$ in entrambi i casi, $\sigma_{\text{BA}} = 0.019$). Il rumore di diffusione dell'errore introdotto dal dithering non altera in misura sufficiente il segnale da compromettere la decodifica, risultando



(a) 256 colori, dithering. (b) 128 colori, dithering. (c) 64 colori, dithering.

(d) 256 colori, senza dithering. (e) 128 colori, senza dithering. (f) 64 colori, senza dithering.

Figura 3.14: PNG Quantization con algoritmo median cut: dithering Floyd-Steinberg attivo (riga superiore) e disattivo (riga inferiore) per 256, 128 e 64 colori.

trasparente rispetto alla rilevazione del watermark. Con soli 64 colori la posterizzazione nelle zone a bassa varianza è visibile, ma il segnale watermark, distribuito sull'intera immagine, rimane intatto.

Pipeline sequenziale

Come caso di stress test estremo, è stato testato un attacco composto dall'applicazione *in sequenza* di sette trasformazioni selezionate sulla base dei risultati ottenuti dai 47 attacchi precedenti:

$$\texttt{GammaCorr. 0.85} \rightarrow \texttt{ColorJitter} \rightarrow \texttt{ChromaQuant32} \rightarrow \texttt{HSV\_Quant} \rightarrow \texttt{PNGQuant 64D} \rightarrow \texttt{JPEG q30} \rightarrow \texttt{WebP q40}$$

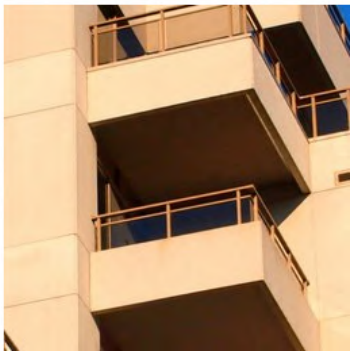
La Figura 3.15 visualizza l'effetto cumulativo delle trasformazioni passo per passo, mentre la Figura 3.16 confronta le distribuzioni di BA per i cinque attacchi più significativi dell'intero esperimento.

La pipeline sequenziale produce una BA media di $\mu_{\text{BA}} = 0.836$ con deviazione standard $\sigma_{\text{BA}} = 0.097$ e $\%\,\text{WM} = 93\%$: **7 immagini su 100 perdono il watermark** nonostante la qualità visiva dell'immagine rimanga accettabile lungo tutti gli 8 step. Questo risultato dimostra che la combinazione di trasformazioni eterogenee non determina un degrado semplicemente riconducibile alla somma degli effetti delle singole trasformazioni: la maggior parte delle trasformazioni della pipeline appartiene alle categorie con robustezza totale (fotometriche, quantizzazione), suggerendo che il contributo predominante al degrado finale derivi dagli ultimi due passaggi di compressione lossy.

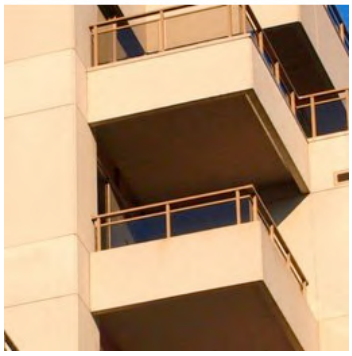
Sintesi

I risultati dimostrano che PixelSeal è un sistema di watermarking robusto nei confronti della quasi totalità delle trasformazioni tipiche del post-processing fotografico e della distribuzione web. La Tabella 3.2 riassume per macrocategoria i valori estremi di robustezza.

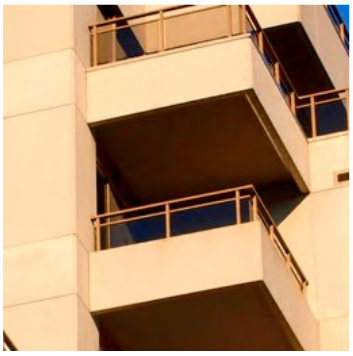
Su 47 attacchi, solamente **tre** causano una riduzione della percentuale di watermark residuo superiore al 5%:



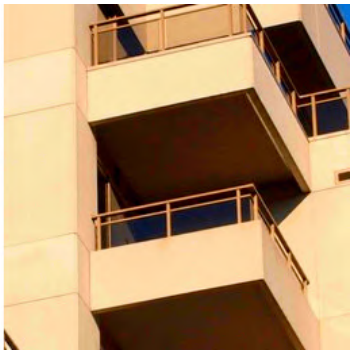
(a) Riferimento con watermark.



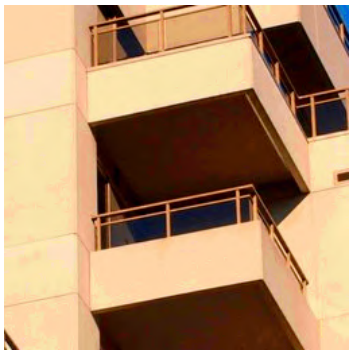
(b) + Gamma 0,85.



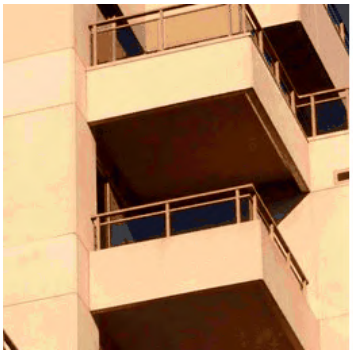
(c) + Color Jitter.



(d) + Chroma Quant. 32.

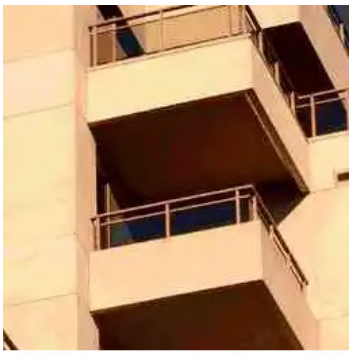


(e) + HSV Quant.



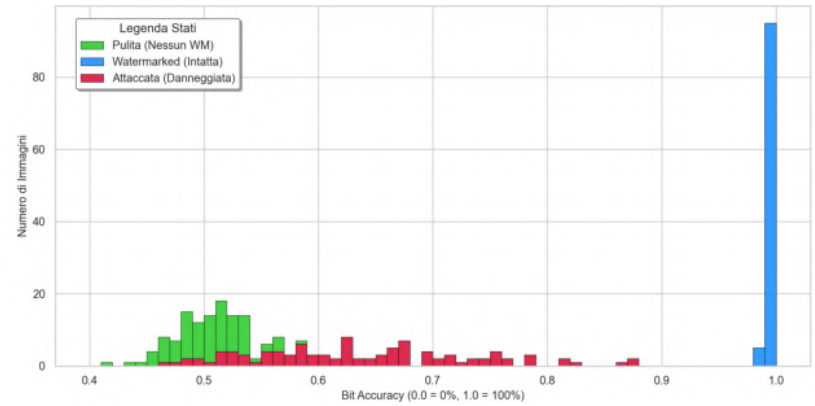
(f) + PNG Quant. 64D.

(g) + JPEG $q30$.

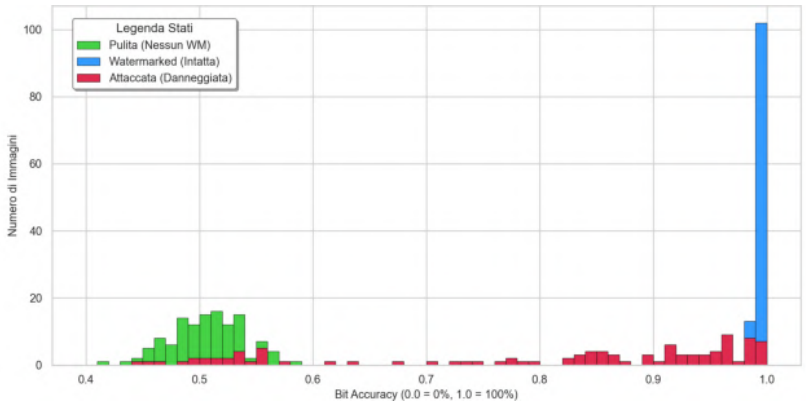


(h) + WebP $q40$: risultato finale ($\mu_{\mathrm{BA}} = 0{,}836$).

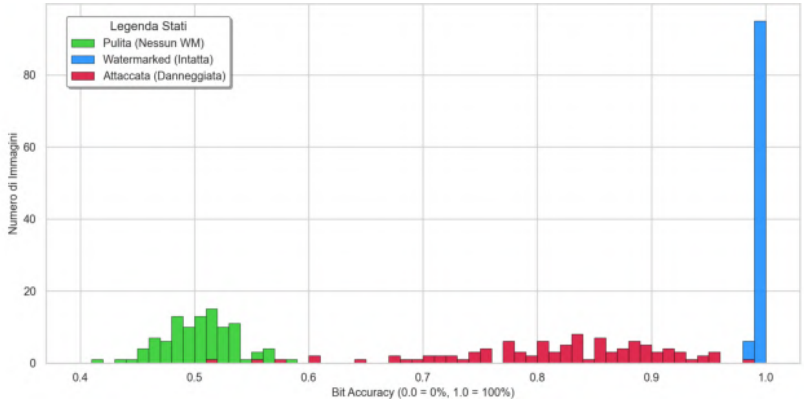
Figura 3.15: Pipeline sequenziale: effetto cumulativo delle sette trasformazioni. Ogni pannello mostra il risultato dopo l'applicazione di tutte le operazioni fino a quel punto.



(a) AWGN $\sigma = 0{,}20$.

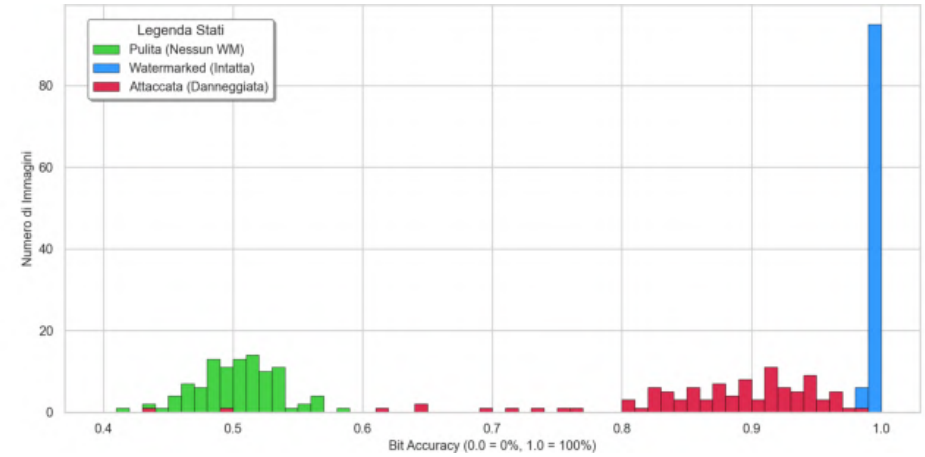


(b) Motion Blur diagonale.

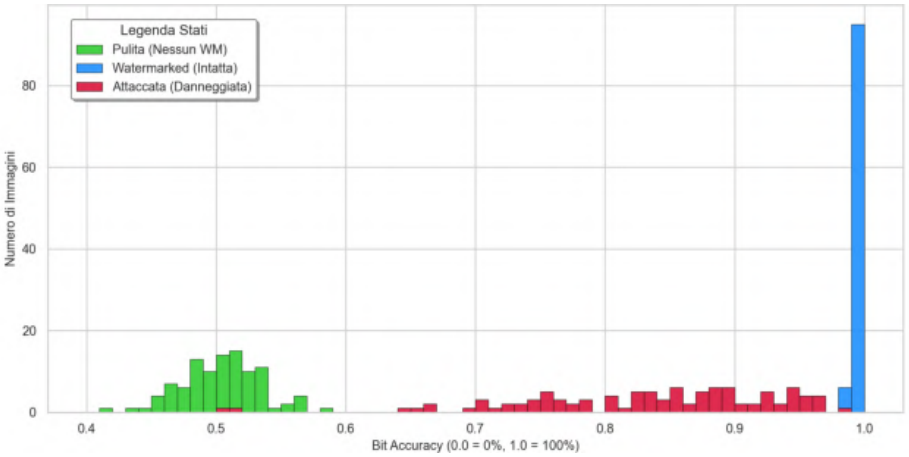


(c) AWGN $\sigma = 0{,}10$.

Figura 3.16: Distribuzioni di bit accuracy per i cinque attacchi più impattanti. In ogni istogramma: immagini pulite (verde), con watermark intatto (blu) e attaccate (rosso). *(Continua nella figura seguente.)*



(d) Center Crop 50%.



(e) Pipeline sequenziale.

Figura 3.16: (*segue*) Center crop e pipeline sequenziale.

Tabella 3.2: Riepilogo della robustezza di PixelSeal per macrocategoria. Per ogni categoria sono riportati il numero di attacchi, la Bit Accuracy (BA) minima osservata e la percentuale di watermark residuo.

Macrocategoria	**N.**	$\min(\mu_{\text{BA}})$	$\min(WM_{residuo})$
Compressione	6	0.947	99%
Rumore	4	0.638	25%
Blur / Nitidezza	8	0.810	74%
Crop / Scale	5	0.871	94%
Upscale $\rightarrow$ Downscale	4	0.998	100%
Variazione colore	7	0.994	100%
Perturbazione LSB	3	0.965	100%
Spazio colore alt.	2	0.986	100%
Geometrica	2	0.997	100%
Quant. PNG	6	0.984	100%
Pipeline seq.	1	0.836	93%

1. **AWGN** σ=0.20 (% WM = 25%): l'attacco con maggiore impatto sulla bit accuracy, dove il rumore broadband ad alta ampiezza supera l'energia del segnale watermark.

2. **AWGN** σ=0.10 (% WM = 90%): degrado significativo ma non critico, già al limite della percezione visiva umana.

3. **Motion Blur Diagonale** (% WM = 74%, $\sigma_{\text{BA}} = 0.179$): attacco ad alta variabilità, con comportamento fortemente dipendente dalla struttura spaziale dell'immagine.

Le categorie più sicure, come Upscale, Variazioni colore, LSB, Spazio colore alternativo, Geometrica, raggiungono una μ_{BA} minima superiore a 0.965 e una percentuale di watermark pari al 100% su tutti gli attacchi della categoria. Anche la pipeline sequenziale con 7 trasformazioni combinate preserva il watermark nel 93% dei casi, confermando la robustezza del modello in scenari di degrado multiplo e realistico.

3.2.2 Attacco a PixelSeal mediante denoising neurale

Si confrontano qui i due approcci introdotti nel Capitolo 2: l'attacco basato sulla sola U-Net di denoising senza detector (Esperimento 1, loss L_1) e quello che affianca alla rete il detector PixelSeal congelato (Esperimento 2, loss MSE con termine *adversarial* $\mathcal{L}_{\text{adv}}$). La valutazione è condotta sull'insieme di test di 100 patch 256×256 (corrispondenti a 25 immagini originali, ciascuna suddivisa in quattro ritagli indipendenti secondo la procedura descritta nella Sezione 2.4.3). Per ciascuna immagine si registrano la bit accuracy nei tre stati: *pulita*, per stabilire il livello di riferimento inferiore del detector in assenza di qualsiasi segnale; *con watermark*, per verificare che il payload sia integralmente recuperabile prima dell'attacco; e *attaccata*, che costituisce la misura principale dell'efficacia della rimozione.

Confronto quantitativo dei due approcci

La Tabella 3.3 riassume le metriche aggregate sui due esperimenti. La bit accuracy media sulle immagini con watermark è pari a 1.00 in entrambi i casi, a conferma che il detector recupera integralmente il payload prima dell'attacco; sulle immagini pulite si attesta intorno a 0.50, ossia al livello di puro caso.

Tabella 3.3: Risultati aggregati dei due approcci di attacco a PixelSeal sull'insieme di test ($n = 100$ patch). La bit accuracy media sulle immagini pulite è ≈ 0.50 e sulle immagini con watermark intatto è 1.00 per entrambi gli approcci; la colonna *BA attaccata* riporta il valore dopo l'applicazione dell'attacco. Il drop è calcolato come differenza tra BA con watermark intatto e BA attaccata. Il PSNR misura la fedeltà della ricostruzione rispetto all'immagine pulita originale.

Approccio	**Loss**	**BA attaccata (media)**	**Drop medio**	**Dev. std BA**	**PSNR medio**
U-Net senza detector	L_1	0.972	0.028	0.042	26.6
U-Net con detector	MSE	0.900	0.100	0.068	22.6

L'introduzione del detector incrementa sensibilmente la capacità di rimozione: il calo medio di bit accuracy rispetto all'immagine con watermark intatto passa da 0.028 (Esperimento 1) a 0.100 (Esperimento 2), con una distribuzione più dispersa (deviazione standard da 0.042 a 0.068). Ciò avviene però a scapito della fedeltà visiva: il PSNR tra immagine ricostruita e immagine pulita originale scende da 26.6 a 22.6. La Figura 3.17 visualizza questa differenza: la distribuzione della

bit accuracy residua dell'approccio con detector è spostata verso valori inferiori e presenta una coda più marcata verso il livello di casualità.

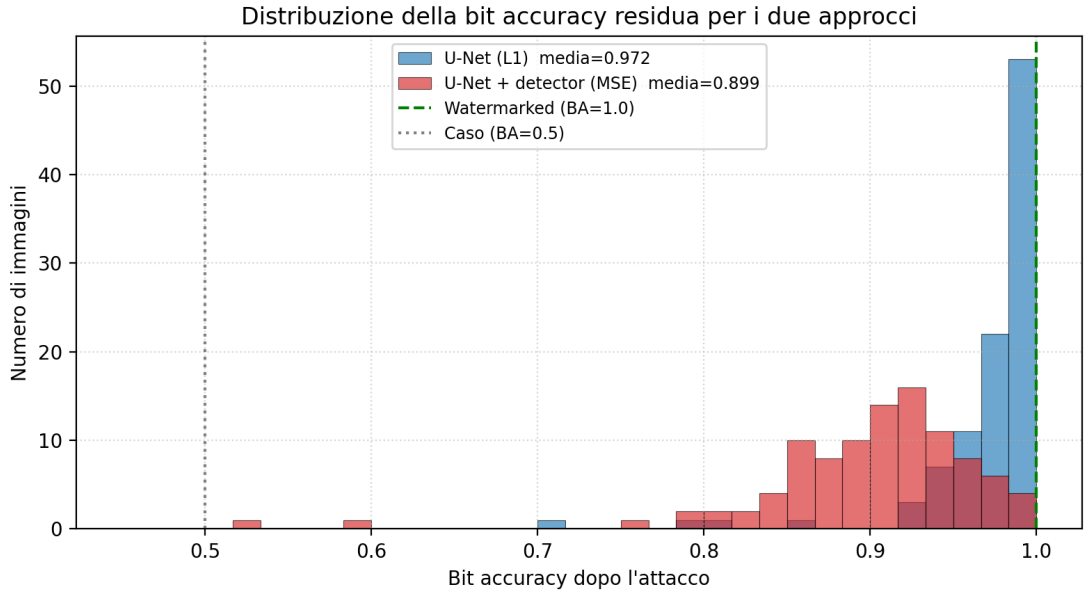


Figura 3.17: Distribuzione della bit accuracy residua dopo l'attacco per i due approcci. L'approccio con detector (in rosso) sposta la massa verso valori più bassi rispetto alla sola U-Net (in blu); la linea verde indica il valore delle immagini con watermark intatte (1.0), quella grigia il livello di puro caso (0.5).

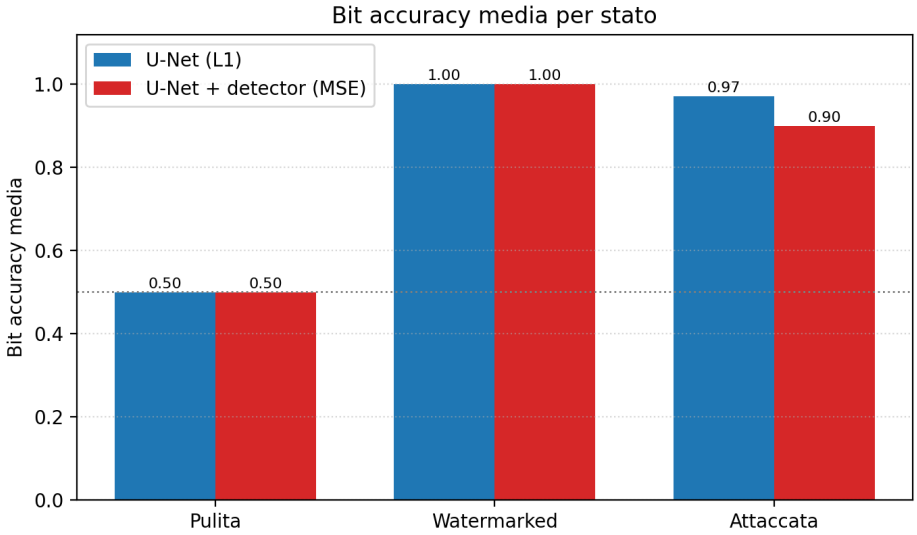


Figura 3.18: Bit accuracy media nei tre stati per i due approcci. In entrambi i casi il watermark intatto è perfettamente decodificato (1.0) e l'immagine pulita si colloca al livello di caso (0.5); lo stato attaccato evidenzia la maggiore efficacia di rimozione dell'approccio con detector.

Analisi qualitativa

La Figura 3.19 riporta tre esempi rappresentativi, selezionati tra i casi di maggiore efficacia dell'attacco con detector. Per ciascuno si mostrano, nell'ordine: immagine originale, immagine con watermark, residuo del watermark amplificato, e le ricostruzioni prodotte dai due attacchi. L'approccio con sola U-Net (L_1) preserva quasi integralmente la qualità visiva (PSNR elevato) ma incide poco sul payload, con la bit accuracy che rimane prossima a 1.0. L'approccio con detector (MSE) abbatte la bit accuracy in modo più deciso, fino a 0.52 nel primo esempio, prossimo al livello di puro caso, al prezzo di una leggera alterazione cromatica e riduzione della nitidezza.

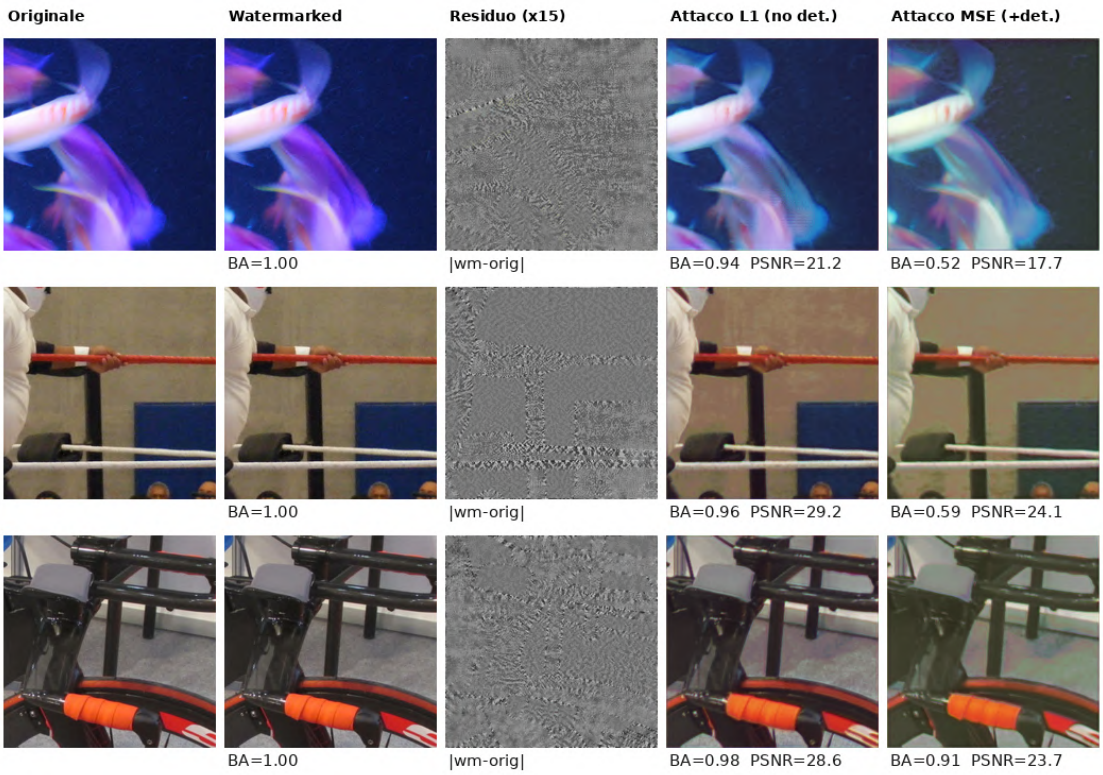


Figura 3.19: Confronto qualitativo dei due attacchi su tre immagini di test. Colonne: originale, con watermark, residuo del watermark amplificato ($\times 15$), ricostruzione con sola U-Net (L_1), ricostruzione con U-Net e detector (MSE). Sotto ogni ricostruzione sono riportate la bit accuracy residua e il PSNR rispetto all'originale.

Stabilità dell'addestramento

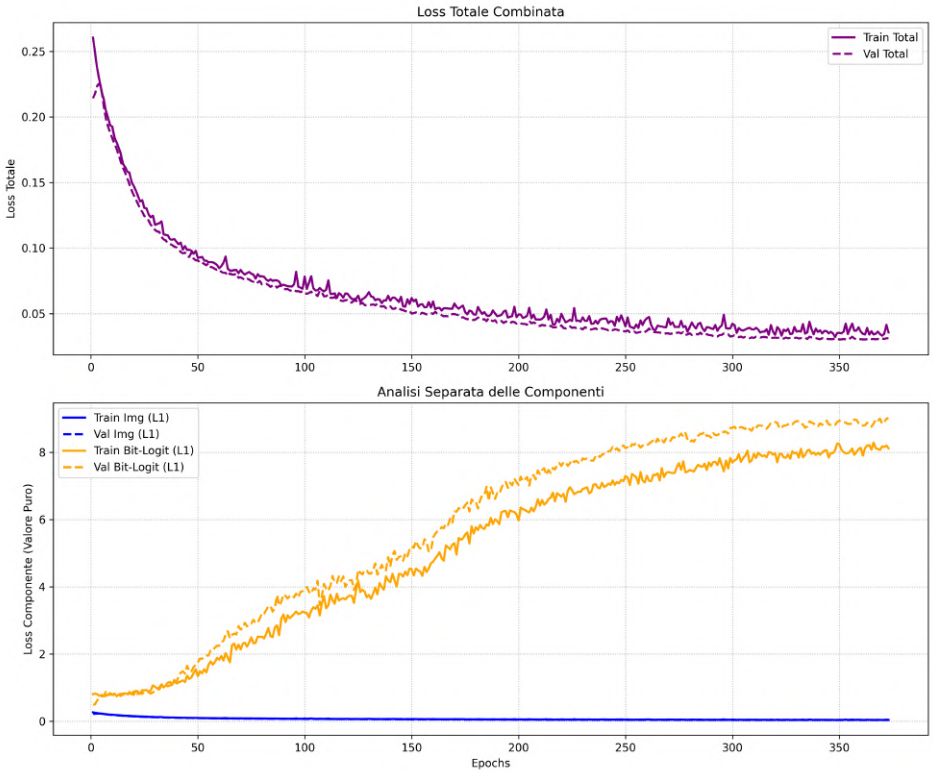
La Figura 3.20 mostra le loss separate dei due addestramenti. L'Esperimento 1 (U-Net senza detector, loss L_1) converge in modo regolare: la loss decresce monotonicamente nelle prime epoche e si stabilizza intorno all'epoca 200, con le curve di training e validation sostanzialmente sovrapposte, segno di assenza di overfitting marcato.

L'Esperimento 2 (U-Net con detector, loss MSE con termine *adversarial*) mostra un comportamento diverso: la loss totale cresce durante le prime 200 epoche, in corrispondenza del warmup lineare di w_{adv}, per poi stabilizzarsi su un plateau. Questo andamento crescente non indica instabilità, ma riflette la natura del termine *adversarial*: man mano che w_{adv} aumenta, la loss combinata incorpora un

contributo crescente di $\mathcal{L}_{\text{adv}}$, che non converge verso zero: il suo obiettivo è portare la probabilità media dei bit verso 0.5, un valore di equilibrio e non di minimo assoluto. Il gap persistente tra la curva di validation e quella di training nella seconda metà dell'addestramento suggerisce che la rete fatica a generalizzare il segnale *adversarial* oltre il set di training, il che è coerente con la limitata dimensione del dataset e con la difficoltà intrinseca di distruggere un residuo additivo distribuito sull'intera immagine tramite un segnale *adversarial* scalare.

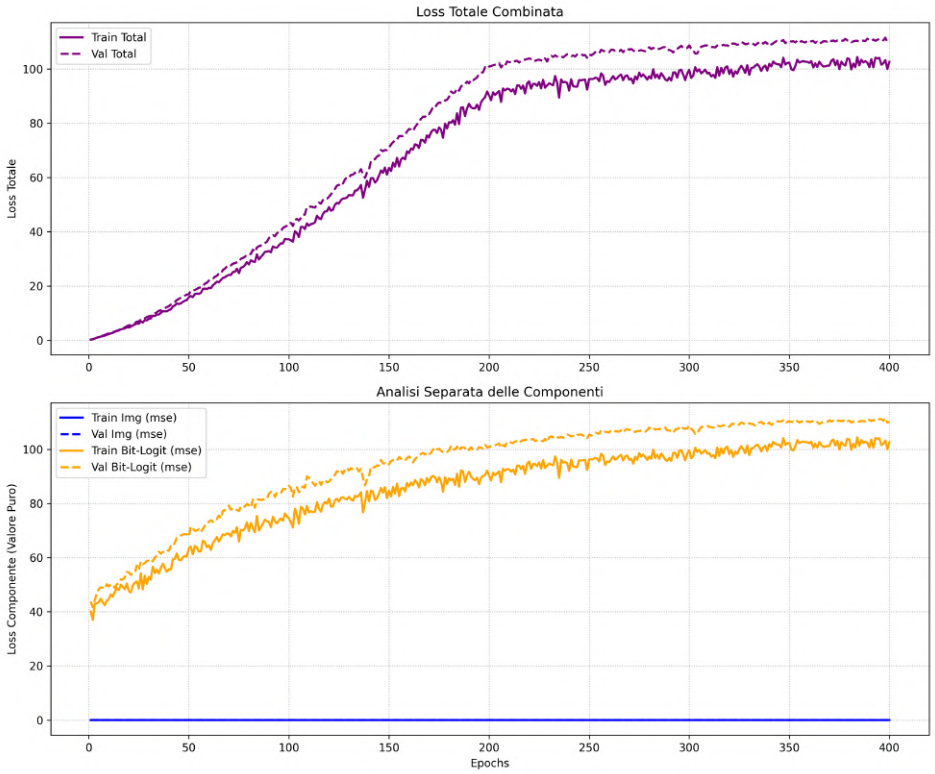
Analisi dei risultati

I risultati delineano un quadro coerente. Da un lato, la sola ricostruzione verso l'immagine pulita, l'approccio senza detector, non è sufficiente a rimuovere il watermark di PixelSeal, che sopravvive quasi intatto al denoising, con un calo di bit accuracy trascurabile. Dall'altro, l'introduzione del segnale *adversarial* rende l'attacco effettivamente più incisivo, ma il guadagno è contenuto: la BA media dopo l'attacco scende da 0.972 (Esperimento 1) a 0.900 (Esperimento 2), senza avvicinarsi al livello di puro caso (BA = 0.5) né alla soglia critica di decodifica (BA = 0.70), al di sotto della quale il payload non è più recuperabile in modo affidabile [15]. Questo risultato suggerisce che la loss *adversarial* scalare adottata, che penalizza il bilanciamento medio dei bit sull'intero batch, non sia sufficiente a distruggere un segnale distribuito su 256 bit indipendenti: abbassare la media non implica rendere ogni singolo bit indistinguibile dal caso. Un termine *adversarial* che agisse bit per bit, o che sfruttasse direttamente il gradiente rispetto alla BA, potrebbe risultare più efficace, al costo di una maggiore instabilità di addestramento. Nel complesso, PixelSeal si conferma marcatamente robusto a questa famiglia di attacchi appresi, in linea con il posizionamento del modello sulla frontiera robustezza e impercettibilità.



(a) Esperimento 1: U-Net senza detector (loss L_1).

Figura 3.20: Curve di loss per i due esperimenti. Per ciascuno: pannello superiore = loss totale combinata (training e validation); pannello inferiore = componenti separate (termine di fedeltà in blu, termine *adversarial* in arancione). *(Continua nella figura seguente.)*



(b) Esperimento 2: U-Net con detector (loss MSE con termine *adversarial*).

Figura 3.20: (*segue*) Esperimento 2.

Conclusioni

Questa tesi ha analizzato la robustezza di due sistemi di watermarking neurale per immagini generate da intelligenza artificiale, *SynthID* di Google DeepMind e *PixelSeal* di Meta, nei confronti di tecniche di rimozione eterogenee, che spaziano da approcci puramente basati sull'elaborazione del segnale fino ad attacchi appresi mediante reti neurali.

Per SynthID, la pipeline di rimozione spettrale *reverse-SynthID* V4 ha prodotto risultati asimmetrici tra le due categorie di immagini testate. Sulle immagini *image+text-to-image* (IT2I) la rimozione ha avuto successo nel 29% dei casi, mentre sulle immagini *text-to-image* (T2I) l'efficacia si è attestata all'1%. L'asimmetria osservata potrebbe dipendere da tre fattori concomitanti: la maggiore eterogeneità di risoluzione delle IT2I, che può rendere il codebook spettrale meno calibrato; la possibile interferenza del processo di condizionamento sull'uniformità della firma; e l'influenza delle caratteristiche visive dell'immagine di partenza sulla qualità del segnale incorporato. Tuttavia, il watermark di SynthID si è dimostrato ampiamente resistente, permanendo nella grande maggioranza dei casi. Un limite importante di questa valutazione è l'assenza di accesso al detector ufficiale di Google: la verifica dell'esito della rimozione ha richiesto di sottoporre ciascuna immagine attaccata al sistema di verifica SynthID tramite Gemini, rendendo la raccolta dei dati onerosa e dipendente da un servizio esterno. Una caratterizzazione più solida richiederebbe l'accesso diretto al detector, la variazione del preset di intensità dell'attacco e un dataset di valutazione più ampio.

La valutazione di PixelSeal su una batteria di 48 attacchi tradizionali ha rivelato un sistema di watermarking marcatamente robusto alla quasi totalità delle trasformazioni tipiche del post-processing fotografico e della distribuzione digitale. In 40 casi su 48 la percentuale di immagini con watermark ancora rilevabile rimane

pari al 100%, mentre in altri cinque scende solo lievemente, restando comunque prossima alla saturazione. Le uniche eccezioni rilevanti riguardano tre attacchi, per i quali la percentuale di immagini con watermark ancora rilevabile scende rispettivamente al 90% (AWGN a intensità media, $\sigma = 0.10$), al 74% (motion blur diagonale, con elevata variabilità dipendente dalla struttura spaziale dell'immagine) e al 25% (rumore gaussiano ad alta intensità, AWGN $\sigma = 0.20$). Anche la pipeline sequenziale composta da 7 trasformazioni eterogenee, progettata per massimizzare l'effetto cumulativo, preserva il watermark nel 93% delle immagini. Questi risultati sono coerenti con l'architettura di PixelSeal: il residuo additivo appreso, attenuato dalla maschera percettiva JND, distribuisce il segnale in modo ridondante su componenti spaziali dell'immagine meno sensibili alle comuni trasformazioni di post-processing.

L'attacco mediante rete U-Net di denoising ha evidenziato i limiti di un approccio puramente ricostruttivo. L'Esperimento 1 (loss L_1) produce ricostruzioni visivamente fedeli (PSNR 26.6) ma non rimuove efficacemente il watermark, con un calo di bit accuracy medio di appena 0.028. L'introduzione del termine *adversarial* nell'Esperimento 2 (loss MSE) incrementa sensibilmente l'efficacia della rimozione, con un calo di bit accuracy medio di 0.100, al costo di una riduzione della fedeltà visiva (PSNR 22.6). In nessuna delle due configurazioni la bit accuracy media scende vicino al livello di puro caso (0.5) né alla soglia critica di decodifica (0.70): anche l'approccio migliore si ferma a una media di 0.900. La loss *adversarial* scalare adottata, che penalizza lo scostamento della probabilità media dei bit del watermark estratti dal detector dal valore di massima incertezza (0,5), non è sufficiente a neutralizzare un segnale distribuito su 256 bit indipendenti. Un termine *adversarial* che agisca bit per bit, o che sfrutti direttamente il gradiente rispetto alla bit accuracy, potrebbe risultare più efficace, al costo di una maggiore complessità di addestramento.

Nel complesso, i due sistemi analizzati mostrano profili di robustezza distinti ma entrambi elevati. SynthID, basandosi su una codifica nel dominio delle frequenze, risulta quasi impenetrabile all'attacco spettrale sulle immagini text-to-image, mentre mostra una vulnerabilità parziale su quelle condizionate. PixelSeal, fondandosi su un residuo additivo appreso e distribuito, resiste efficacemente sia agli attacchi tradizionali sia a quelli neurali, pur mostrando una riduzione della

robustezza sotto pressione *adversarial* diretta.

I risultati suggeriscono che gli attuali attacchi non siano ancora sufficienti a compromettere sistematicamente i watermark neurali di nuova generazione. Le tecniche considerate riescono a indebolire il segnale senza eliminarlo completamente e, quando aumentano la probabilità di rimozione, introducono un degrado visivo proporzionalmente crescente. Questo equilibrio tra efficacia di rimozione e qualità dell'immagine indica che i sistemi di watermarking analizzati hanno consolidato una robustezza che non è ancora stata superata dalle tecniche di attacco disponibili, lasciando aperto il problema della rimozione senza perdita percettiva come direzione centrale per la ricerca futura.

Sviluppi futuri potrebbero esplorare termini avversariali bit-wise per l'attacco a PixelSeal, l'uso di campioni più ampi e di preset di intensità variabili per SynthID, e l'estensione dell'analisi ad altri sistemi di watermarking emergenti. Parallelamente, sarebbe interessante valutare la robustezza di questi sistemi in scenari di distribuzione realistica, dove le immagini attraversano pipeline di compressione e ridimensionamento automatico tipiche delle piattaforme di condivisione, nonché estendere il confronto a nuovi approcci al watermarking che verranno introdotti nei futuri modelli generativi.

Ringraziamenti

Ringrazio il relatore, Professor Matteo Ferrara, e i correlatori, Dottor Serafino Pandolfini e Dottor Lorenzo Pellegrini, per il supporto nello sviluppo del progetto e nella stesura della tesi.

Ringrazio la mia famiglia, la mia ragazza e i miei amici che mi hanno accompagnato lungo tutto questo percorso.

Bibliografia

[1] MathWorks. Autocodificatori (autoencoders). `https://it.mathworks.com/discovery/autoencoder.html`, 2026. Ultimo accesso: 10 giugno 2026.

[2] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, November 2019.

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

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

[5] Google Developers. Gan structure. `https://developers.google.com/machine-learning/gan/gan_structure`, 2026. Ultimo accesso: 10 giugno 2026.

[6] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

[8] Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann, 2nd edition, 2007.

[9] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models, 2023.

[10] Jiaming Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding digital data in images via end-to-end optimization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 757–773, 2018.

[11] Ahmad Rezaei, Mohammad Akbari, Saeed Ranjbar Alvar, Arezou Fatemi, and Yong Zhang. Lawa: Using latent space for in-generation image watermarking, 2025.

[12] Google DeepMind. Synthid - a tool to watermark and identify content generated through ai, 2023.

[13] Timothy O'Keefe, Pierre Fernandez, Hervé Jégou, and Google DeepMind. Synthid-image: Image watermarking at internet scale. *arXiv preprint arXiv:2503.02345*, 2025.

[14] fyx.me. Attempting model extraction of google deepmind synthid (image) watermark detector and exploring adversarial machine learning attacks against synthid, 2026.

[15] Tomáš Souček, Pierre Fernandez, Hady Elsahar, Sylvestre-Alvise Rebuffi, Valeriu Lacatusu, Tuan Tran, Tom Sander, and Alexandre Mourachko. Pixel seal: Adversarial-only training for invisible image and video watermarking, 2025.

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023.

[17] Alexander Nemecek, Hengzhi He, Guang Cheng, and Erman Ayday. Authenticated contradictions from desynchronized provenance and watermarking, 2026.

[18] OpenAI. C2pa in DALL·E 3, 2024.

[19] Coalition for Content Provenance and Authenticity. C2pa technical specification. Technical report, C2PA, 2024.

[20] OpenAI. Advancing content provenance for a safer, more transparent AI ecosystem. `https://openai.com/index/advancing-content-provenance/`, 2026.

[21] Jessica Young, Sam Vaughan, Andrew Jenks, Henrique Malvar, Christian Paquin, Paul England, Thomas Roca, Juan LaVista Ferres, Forough Poursabzi, Neil Coles, Ken Archer, and Eric Horvitz. Media integrity and authentication: Status, directions, and futures, 2026.

[22] Zhengxin Pan, Yanzhi Guo, Jian Yang, Ting Yao, Chuan Zhang, and Xiaoming Zhang. invismark: Invisible image watermarking resilient to severe real-world degradations. *arXiv preprint arXiv:2411.07795*, 2024.

[23] Qwen Team. Qwen3.5-omni technical report, 2026.

[24] Alosh Denny. reverse-SynthID: An independent reverse-engineered implementation of Google DeepMind's SynthID. `https://github.com/aloshdenny/reverse-SynthID`, 2024.

[25] Pierre Fernandez, Hady Elsahar, I. Zeki Yalniz, and Alexandre Mourachko. Video seal: Open and efficient video watermarking. *arXiv preprint arXiv:2412.09492*, 2024.

[26] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022.