Università degli Studi di Firenze

Scuola di Ingegneria - Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Triennale in Ingegneria Informatica

Identificazione di immagini sintetiche tramite codifica JPEG AI

Candidato
Rustichini Edoardo

Relatore
Prof. Piva Alessandro

Correlatori
Shullani Dasara
Baracchi Daniele

Anno Accademico 2024-2025

Indice

Introduzione

I continui progressi nel campo dell'intelligenza artificiale hanno portato ad avere tecniche per la generazione di contenuti digitali che risultano avere un livello di realismo tale da essere indistinguibili da quelli reali. La diffusione di queste tecnologie al grande pubblico aumenta il rischio di utilizzi illeciti, con potenziali conseguenze negative. Il rilevamento di questo tipo di contenuti è uno dei temi principali della *multimedia forensics*, disciplina che consiste nello studio e analisi di contenuti digitali con il fine di verificarne l'autenticità.
Questo lavoro di tesi triennale si propone di esaminare l'uso della codifica di JPEG AI, un codec di compressione basato su reti neurali, per svolgere task di *fake detection* su immagini di volti; in particolare si vuole verificare se l'encoder di JPEG AI riesca ad estrarre dalle immagini rappresentazioni latenti sufficientemente ricche di informazioni da rilevare se un'immagine sia sintetica o reale.

Struttura relazione La relazione è strutturata come segue: nel capitolo 1 vengono presentati gli aspetti teorici necessari per la comprensione del lavoro, tra cui la creazione e identificazione di immagini sintetiche (sez. 1.1) ed il codec JPEG AI (sez. 1.2); nel capitolo 2 viene presentato la metodologia e gli aspetti più tecnici riguardanti il lavoro svolto, mentre nel capitolo 3 sono spiegati gli esperimenti e i loro risultati. Infine nel capitolo 4 sono tratte le conclusioni.

Capitolo 1

Stato dell'arte

1.1 Immagini fake: rischi, generazione e rilevamento

L'incremento degli studi nel campo dei modelli generativi, ossia tecniche di intelligenza artificiale dedicate alla generazione di dati simili a quelli di addestramento, ha portato allo sviluppo di tecnologie in grado di produrre immagini indistinguibili da quelle reali; la diffusione al pubblico di questi strumenti rende la creazione di immagini sintetiche con alto livello di realismo un'operazione semplice ed accessibile anche a utenti non esperti.
Tra le immagini false si possono distinguere due categorie principali: *cheapfakes*, tecniche di manipolazione meno sofisticate, come *splicing* e *copy-move ecc*, e *deepfakes*, metodi che fanno uso di intelligenza artificiale. Il termine deepfake nasce dalla combinazione di "deeplearning" e "fake" e vuole indicare la manipolazione di contenuti già esistenti o la generazione di nuovi tramite l'utilizzo di modelli di deeplearning; il lavoro si concentrerà su questa categoria. L'uso più comune dei deepfakes riguarda la produzione e rielaborazione di immagini rappresentanti volti umani con applicazioni nel mondo dell'intrattenimento o del marketing; le stesse tecnologie possono però essere utilizzate per fini illeciti come la creazione di profili falsi ma apparentemente

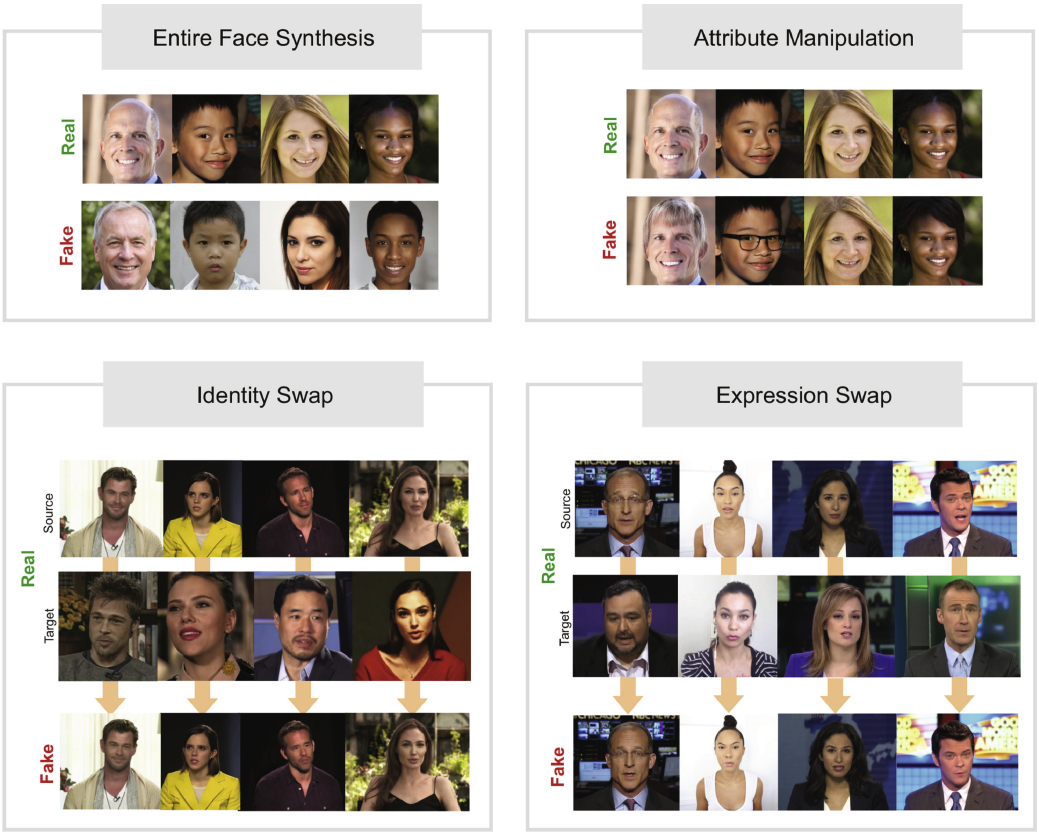


Figura 1.1: Esempi per ogni tipo di manipolazione di volti

autentici, diffusione di disinformazione e produzione di materiale diffamatorio nei confronti di personaggi pubblici o privati.

Manipolazioni di volti Tra le più comuni manipolazioni si trovano [1]: *entire face synthesis*, ossia la creazione di immagini di volti di persone non reali; *identity swap*, che consiste nella sostituzione del volto di un individuo con quello di un altro; *attribute manipulation*, ovvero la modifica di attributi/caratteristiche del volto e *expression swap*, che ha lo scopo di modificare l'espressione facciale della persona scelta.
Questo lavoro si concentra sulla *entire face synthesis* e ci riferiremo principalmente a questa parlando di generazione e identificazione di immagini sintetiche.

1.1.1 Generazione

I modelli generativi hanno l'obbiettivo di apprendere la distribuzione dei dati di addestramento e trovare il miglior modo per rappresentarli in modo da generare nuovi campioni simili a quelli originali. Per la creazione di immagini di volti sintetiche sono usate principalmente tre diverse architetture [2]: Autoencoders, Diffusion Models e GANs. Comprenderne il funzionamento è fondamentale per sviluppare metodi di identificazione delle immagini prodotte.

Autoencoders Sono architetture composte da un encoder, che ha l'obbiettivo di trasformare l'input in una rappresentazione latente compatta, ed un decoder che ricostruisce l'immagine desiderata. Sono spesso usate nell'identity swapping sfruttando un encoder comune che apprende una rappresentazione latente condivisa, e decoder separati che si specializzano nella ricostruzione di immagini relative ad un individuo specifico [2], così da ottenere uno scambio di identità decodificando la rappresentazione compatta della persona A con il decoder della persona B.

Diffusion Models I modelli di diffusione sono modelli generativi il cui funzionamento si basa su due fasi: nel *forward process* un'immagine reale viene corrotta introducendo rumore in un determinato numero di iterazioni, e successivamente nel *reverse process* il modello viene addestrato per sintetizzare un'immagine realistica rimuovendo progressivamente rumore da rumore puro [3].L'approccio iterativo di queste fasi permette di generare immagini di alta qualità riuscendo a catturare dettagli fini; per questo i Diffusion Models rappresentano attualmente lo stato dell'arte nel campo della generazione di immagini.

1.1.2 GANs

GAN (*Generative Adversarial Networks*) è l'architettura proposta nel 2014 da Goodfellow et al. [4] composta da due reti neurali: generatore e discriminatore. Il generatore G riceve in input rumore casuale z e lo mappa in un'immagine sintetica $G(z)$; il discriminatore D riceve in input immagini reali o sintetiche ed ha il compito di classificarle. Le due reti vengono addestrate contemporaneamente competendo in un *min-max game* nel quale G, generando immagini sempre più realistiche, ha l'obbiettivo di massimizzare la probabilità che D compia un errore nella classificazione, mentre D deve minimizzare questa probabilità. L'equazione che descrive questo gioco è la seguente:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \quad (1.1)$$

Durante l'addestramento il gradiente della funzione di perdita di D verrà usato per aggiornare G. Inizialmente G genererà immagini rumorose e facilmente identificabili come falsi da D; con il procedere della fase di addestramento la qualità delle immagini aumenterà fino a diventare abbastanza realistici da ingannare D. Questa architettura ha rappresentato il primo salto di qualità nel campo della generazione di immagini e per anni è stata il punto di riferimento. Di seguito viene approfondita la variante di GAN usata per creare il dataset di immagini sintetiche per gli esperimenti svolti (vedi sez 2.3).

StyleGAN StyleGAN [5] presenta un'importante innovazione rispetto alla struttura classica: il generatore non riceve come input direttamente il rumore casuale z, ma questo viene prima trasformato da un mapping network $f : \mathcal{Z} \to \mathcal{W}$ in un vettore $f(z) = w$ in uno spazio latente intermedio $\mathcal{W}$ non vincolato dalla distribuzione del dataset di addestramento, proprietà invece di $\mathcal{Z}$. Successivamente w viene trasformato in una serie di stili y che sono interpretate da ogni layer del generatore come istruzioni su come operare at-

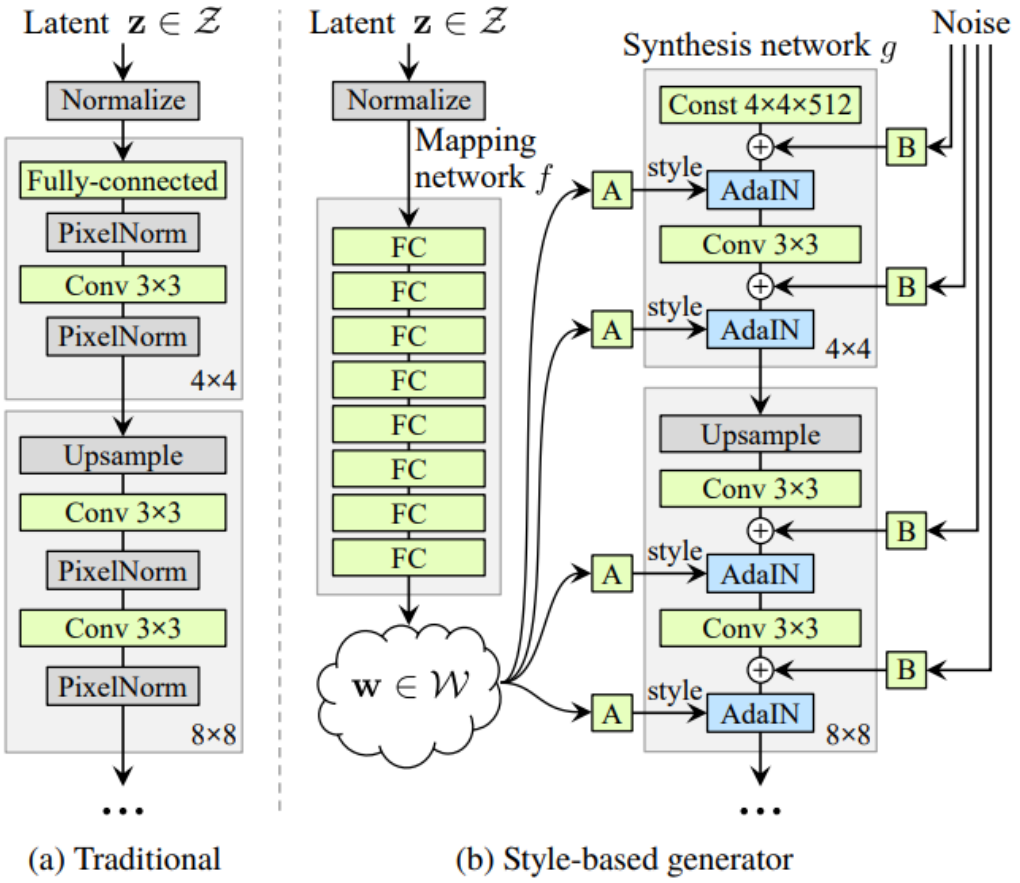


Figura 1.2: Architettura di StyleGAN descritta in [5]

traverso un'operazione chiamata *Adaptive Instance Normalization* (AdaIN), controllando così il contributo di ogni layer alla generazione dell'immagine. L'architettura è visbile in fig. 1.2. L'innovazione di StyleGAN consiste quindi nel modo innovativo di generare immagini basandosi su una collezione di stili; in particolare gli effetti di ogni stile sono localizza nella rete, permettendo di scegliere sottoinsiemi di stili per modificare solo alcuni aspetti dell'immagine. Da notare come StyleGAN non modifica la struttura del discriminatore, che rimane quello classico, ma cambia il controllo e la qualità del generatore, permettendo di ottenere immagini più realistiche e con un maggior controllo sugli attributi generati.

1.1.3 Rilevamento

Il problema del rilevamento dei deepfake è formulato come un problema di classificazione binaria, nel quale, data un immagine I, l'obbiettivo è prevedere se I appartiene alla classe di immagini reali o a quella di immagini sintetiche.

Approcci classici I metodi di rilevamento classici si basano sull'analisi delle immagini in ricerca di anomalie lasciate durante la fase di creazione dell'immagine. I *Physical-based methods* cercano di sfruttare inconsistenze nell'immagine, come illuminazione o il riflesso della luce negli occhi, facendo assunzioni su proprietà del mondo reale. I *Physiological-based Methods* invece investigano l'aspetto semantico delle immagini, analizzando anomalie riguardanti l'asimmetria di attributi dei volti, come diversi colori negli occhi o diversa forma delle pupille.
Altri metodi invece cercano di trovare anomalie nelle caratteristiche "strutturali" delle immagini tra cui: imperfezioni causate dal sistema ottico delle fotocamere nel processo di acquisizione, o processing interno o esterno alla fotocamera.
Una caratteristica studiata è la PRNU (*Photo Response Non-Uniformity*): ogni fotocamera lascia un pattern di rumore specifico che può essere usato come una sorta di "impronta digitale" della fotocamera; l'assenza di questo pattern in una regione dell'immagine è un forte indicatore di manipolazione.
Altri approcci consistono nell'analisi dei metadati

Approcci basati su deeplearning Gli approcci più recenti sfruttano il deep learning, in particolare si usano reti neurali in modo da estrarre le feature rilevanti delle immagini in modo automatico, senza doverle progettare a mano.
Nel caso di immagini generate da GAN, i classificatori vengono allenati sulla rappresentazione nel dominio della frequenza, dato che è stato dimostrato

che le immagini sintetiche presentano delle anomalie in questo dominio. Anche le CNN vengono usate Gli approcci classici riescono ad ottenere buoni risultati su immagini non compresse, ma questa efficienza diminuisce drasticamente quando si lavora con immagini compresse, come quelle condivise sui social media, caso in cui invece i metodi basati su deeplearning riescono a mantenere un buon livello di accuratezza.

1.2 JPEG AI

La qualità e la risoluzione delle immagini è in continua crescita, e così la loro dimensione in formato non compresso; considerando anche l'aumento del numero di immagini prodotte e visualizzate quotidianamente, si è sviluppata la necessità di trovare dei nuovi metodi di compressione in modo da consentire una trasmissione e un'archiviazione più efficienti. Un nuovo promettente paradigma di compressione è chiamato"*Learning-based image compression*" o "*Neural image compression*" e sfrutta l'apprendimento automatico facendo leva sulle reti neurali per raggiungere livelli di compressione maggiori.

1.2.1 Compressione di immagini

"*A picture is worth a thousand words*"

La compressione dei dati è un problema ampiamente studiato nell'informatica: l'obiettivo è quello di rappresentare un'informazione usando un numero ridotto di bit rispetto alla rappresentazione originale. Nel caso delle immagini questo equivale a rappresentare la stessa informazione visiva, almeno apparentemente, riducendo lo spazio di archiviazione utilizzato. La compressione si divide in due approcci: lossless, senza perdita di informazione, e lossy, ovvero con perdita di informazione.
La compressione senza perdita viene utilizzata nelle applicazioni in cui è richiesta una perfetta ricostruzione dell'immagine originale, per esempio nel caso di immagini mediche. Quella con perdita viene usata per esempio nella condivisione delle immagini sul web perché consente di raggiungere un maggior livello di compressione scartando parte dell'informazione ma mantenendo una qualità accettabile.

Compressione lossy classica Tradizionalmente il processo per la compressione lossy prevede i seguenti passaggi: inizialmente viene applicata una trasformata all'immagine che mappa i pixel dal dominio spaziale ad uno più

efficiente per la compressione, convertendoli in coefficienti. Questi vengono successivamente quantizzati, operazione nella quale avviene la principale perdita di informazione. Infine si effettua una codifica dell'entropia lossless per trasformare la sequenza di coefficienti quantizzati in un bitstream finale.
Il più importante e diffuso esempio di compressione lossy è JPEG [6]; il suo successo è dovuto al basso costo computazionale necessario per il calcolo della DCT (*Trasformata discreta del coseno*), che, applicata a blocchi di 8×8 pixel, mappa i pixel dal dominio spaziale a quello della frequenza, permettendo di concentrare la maggior parte dell'informazione rilevante in pochi coefficienti. Dopo questa trasformata i coefficienti vengono quantizzati in base alla loro frequenza, usando maggiore precisione per le frequenze più basse, che contengono più informazione per l'occhio umano, e approssimando maggiormente quelle più alte.
In generale le tecniche tradizionali sono state progettate "a mano" da ingegneri e ricercatori sfruttando la conoscenza della struttura probabilistica dell'informazione ed infatti fanno uso di una trasformata predeterminata e immutabile, avendo lo svantaggio di non essere abbastanza flessibile e ottimale per tutti i tipi di immagine.

Learned image compression

In questo nuovo paradigma di compressione vengono utilizzate le reti neurali: i primi articoli che propongono tecniche di questo tipo risalgono già agli anni '90 [7, 8], quando però la loro implementazione non era possibile nella pratica. Recentemente, con i progressi nell'apprendimento automatico e lo sviluppo di hardware specializzato come le GPU, anche il campo del learned image compression si è evoluto.
La differenza principale con i codec tradizionali come JPEG risiede nel fatto che in questo nuovo approccio ogni componente classica (trasformata, quantizzazione, codifica dell'entropia) è sostituita da una rete neurale, portando alla creazione di un *learned image codec*.

Architettura La maggior parte delle tecniche sono basate sugli autoencoder [9], un tipo di rete neurale in grado di apprendere la mappatura tra l'input e uno spazio di una rappresentazione latente compatta, spesso chiamato *bottleneck*.
L'architettura che ha riscosso più successo è quella proposta da [10] che ha dato una prova dell'efficienza di questa tecnica, migliorata successivamente da [11].
Uno degli aspetti fondamentali è l'approccio end-to-end nel quale tutti i moduli, dall'encoder al decoder, vengono addestrati insieme mediante un'unica *global loss-function* permettendo un'ottimizzazione "unificata" che massimizza l'efficienza di compressione. Questo rappresenta un vantaggio rispetto ai metodi tradizionali dove invece ogni componente veniva ottimizzata indipendentemente.
Oltre agli autoencoder, si possono trovare anche altri tipi di architettura, e come proposto in [12] si possono classificare in base a questi criteri:

- tipologia di rete neurale utilizzata: oltre a quelli già citati, sono usate anche RNN, per elaborare l'immagine in modo sequenziale, oppure GAN (descritti in sez. 1.1.2)
- dimensione dell'unità di codifica: alcune tecniche scelgono di elaborare l'intera immagine, mentre altre decidono di lavorare a blocchi di pixel, riducendo la complessità
- strategia del controllo del bit rate: alcuni metodi scelgono di usare lo stesso modello per comprimere a più bitrate, altri invece utilzzano diversi modelli ottimizzati per un singolo livello di qualità

1.2.2 Motivazioni per lo sviluppo di JPEG AI

La principale motivazione dietro lo sviluppo è descritta nel seguente modo:

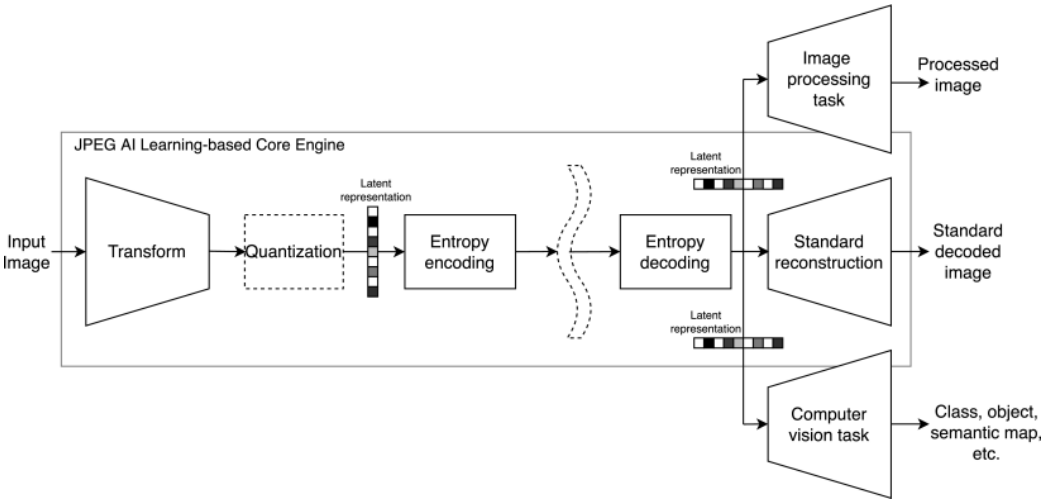
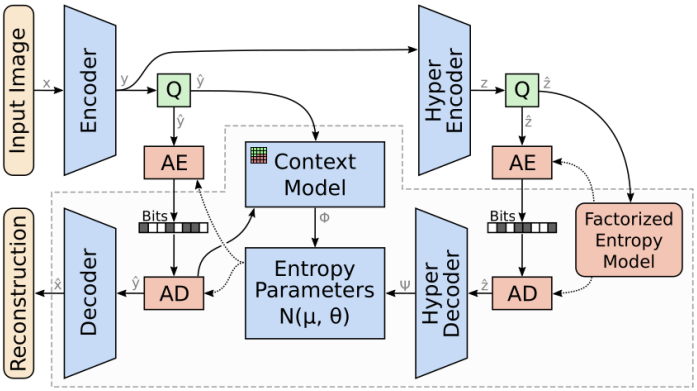


Figura 1.3: Framework JPEG AI

> "*The scope of JPEG AI is to create a learning-based image coding standard that provides a single-stream, compact compressed domain representation targeting both human visualization,.., and effective performance for image processing and computer vision tasks, with the goal of supporting royalty-free baseline* [13]"

L'obbiettivo può essere paragonato alla creazione di un *"linguaggio comune"* che consenta una rappresentazione efficiente sia per la visione umana che per macchine. Questa scelta risulta rilevante considerando che i contenuti digitali non sono più visualizzati esclusivamente dagli umani, ma in molte applicazioni, come sistemi di sorveglianza intelligente, sono le macchine i destinatari principali. L'idea è quindi di usare la stessa rappresentazione compatta e ricca di informazione sia per task di *image processing*, che *computer vision*, dove l'obbiettivo è estrarre dall'immagine informazione semantica, oppure potrebbe essere ricostruita (vedi fig. 1.3) per renderla visibile ad un utente umano. Un unico bitstream multitask offre due vantaggi: il primo riguarda la riduzione della complessità necessaria a svolgere task di image processing o computer vision, consentendo di evitare completamente la parte di ricostruzione dell'immagine e di agire direttamente sulla rappresentazione latente.



Component	Symbol
Input Image	$\boldsymbol{x}$
Encoder	$f(\boldsymbol{x};\boldsymbol{\theta}_e)$
Latents	$\boldsymbol{y}$
Latents (quantized)	$\hat{\boldsymbol{y}}$
Decoder	$g(\hat{\boldsymbol{y}};\boldsymbol{\theta}_d)$
Hyper Encoder	$f_h(\boldsymbol{y};\boldsymbol{\theta}_{he})$
Hyper-latents	$\boldsymbol{z}$
Hyper-latents (quant.)	$\hat{\boldsymbol{z}}$
Hyper Decoder	$g_h(\hat{\boldsymbol{z}};\boldsymbol{\theta}_{hd})$
Context Model	$g_{cm}(\boldsymbol{y}_{<i};\boldsymbol{\theta}_{cm})$
Entropy Parameters	$g_{ep}(\cdot;\boldsymbol{\theta}_{ep})$
Reconstruction	$\hat{\boldsymbol{x}}$

Figura 1.4: Architettura descritta in [11]

Il secondo vantaggio riguarda il potenziale incremento nell'accuratezza delle operazioni: essendo JPEG AI stato progettato, ottimizzato ed addestrato per trovare una rappresentazione compatta latente che riesca a contenere l'informazione (anche semantica) contenuta nell'immagine originale, svolgere task direttamente su questa rappresentazione garantirebbe potrebbe garantire prestazioni migliori rispetto ad utilizzare l'immagine lossy ricostruita, specialmente a bitrate bassi.
Per ottenere questo l'encoder di JPEG AI deve generare un bitstream indipendente da tutti task, ossia non ottimizzato per nessun compito specifico, mantenendo però il requisito più importante di un alto livello di compressione.

1.2.3 Architettura

L'architettura di JPEG AI segue lo schema dei learning-based codec [11] osservabile nella fig.1.4; si possono distinguere due moduli. Il modulo principale è rappresenta l'autoencoder, composto dalla coppia *encoder-decoder*. L'encoder apprende una trasformata non-lineare chiamata *Analysis transform*, che mappa un'immagine x in una rappresentazione latente più compatta y. Il decoder apprendende una trasformata non-lineare chiama-

ta*Synthesis transform* che ricostruisce un'approssimazione dell'immagine orginale partendo dalla versione quantizzata $\hat{y}$ della rappresentazione latente. Il secondo modulo corrisponde al *Modello per l'entropia*: composto da *Hyperprior* e *Context-model* apprende la distribuzione di probabilità dei valori dei tensori latenti per svolgere in modo più efficiente la codifica dell'entropia [10]. L'*Hyperprior* è un autoencoder ausiliario composto da un *hyper-encoder*, che estrae da y delle *side-information/hyper-latent* z che vengono quantizzate ($\hat{z}$) ed incluse nel bitstream, ed un *hyper-decoder*, che partendo $\hat{z}$ aiuta a predire la distribuzione dei valori del tensore latente. Il *Context model* utilizzando un approccio autoregressivo, usa il contesto dato dai valori già codificati $\hat{y}_{<i}$ per stimare la distribuzione dei valori.

Fase di Encoding Il processo di encoding di un'immagine si può osservare in 1.5: inizialmente l'analysis transform, una rete neurale composta da layer convoluzionali e attivazioni non lineari, trasforma l'immagine in un tensore latente y; questo viene elaborato dall'hyper-encoder che estrae da y il tensore z, successivamente arrotondato ($\hat{z}$) e compresso nell'ultima parte del bitstream. Per la codifica entropica di y viene usato un hyper-scale decoder, che riceve $\hat{z}$, per generare i parametri, per esempio le varianze, necessarie per descrivere la distribuzione di probabilità dei valori di y. Questi parametri vengono usati per fare la codifica entropica dei residui, mentre un hyper-mean decoder produce una stima della media del tensoe latente, che viene usata nel modulo di Latent Prediction per stimare i valori di mu_y. Sottraendo μ_y al latente originale y, vengono calcolati i residui; dopo esser stati scalati e arrotondati, sono compressi nel bitstream usando me-tANS.

Fase di Decoding Nella fase di decoding (vedi fig. 1.6) avviene il processo inverso: inizialmente viene fatta la decodifica delle informazioni ausiliari $\hat{z}$, che vengono passate all'hyper-decoder (identico a quello della fase di enco-

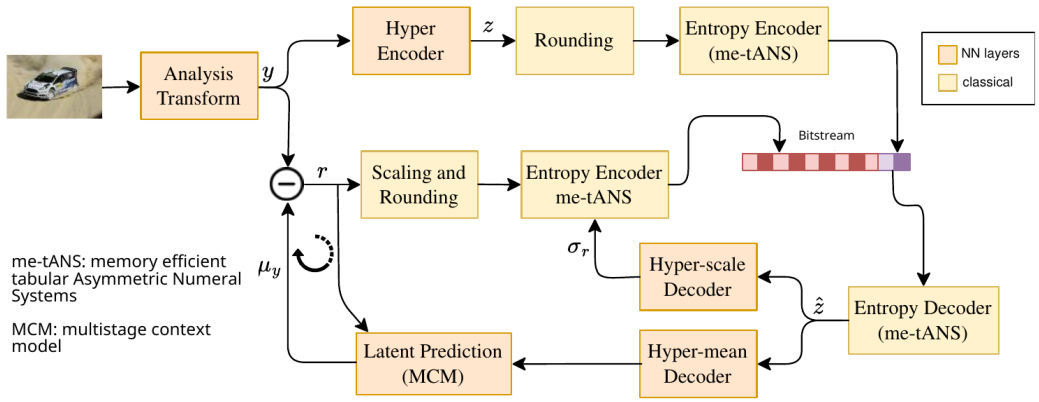


Figura 1.5: Schema della fase di encoding

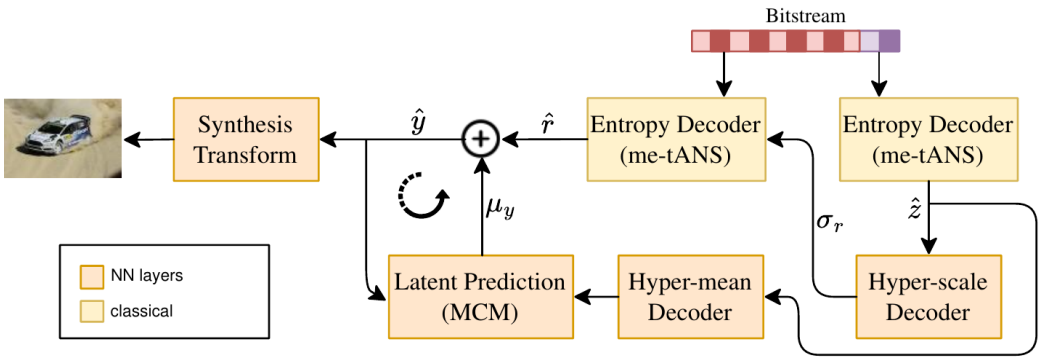


Figura 1.6: Fase di Decoding di JPEG AI

ding); i parametri calcolati da quest'ultimo vengono usati per decodificare in modo più efficiente i residui, ottenndo $\hat{r}$. Attraverso un modulo di latent prediction vengono nuovamente stimati $\hat{y}$, che è ricevuta dalla synthesis transform che produce un'immagine partendo dalla rappresentazione latente ricevuta.

Risultati ottenuti da JPEG AI I primi esperimenti hanno riscontrato risultati promettenti: con le prime versioni delle implementazioni JPEG AI è riuscito a raggiungere un BD-rate gain (*Bjøntegaard-Delta Rate*, una metrica per la valutazione dell'efficienza di compressione) del 28% [13] rispetto a VVC Intra (lo standard attdi compressione più effciente attualmente disponibile)

usando la configurazione più semplice "*tools-off*" senza ottimizzazioni avanzate. Inoltre è stato notato anche un miglioramento nella qualità soggettiva a parità di bitrate [13], confermando il potenziale di questo strumento.

1.2.4 Conseguenze sulla multimedia forensics

La creazione di JPEG AI introduce nuove sfide nel campo della multimedia forensics. In [14] viene definito il termine *miscompression* per indicare gli errori creati nella ricostruzione delle immagini che portano ad avere dettagli semantici diversi tra l'immagine reale e quella ricostruita. Poichè l'architettura è simile a quella dei generatori di immagini sintetici, in [15] viene spiegato come le immagini compresse con JPEG AI presentano artefatti simili a quelle false, che porta i detetctor esistenti classificarli in modo errato rilevandole come immagini sintetiche. In [16] vengono proposti tre indizi forensi per il rilevamento e l'analisi di immagini compresse con JPEG AI: correlazioni di colore, quantizzazione nello spazio latente e analisi rate-distortion.

Capitolo 2

Framework

2.1 JPEG AI Verification Model

Lo strumento utilizzato per l'estrazione delle feature è *jpeg-ai-reference-software* [17]; questo rappresenta l'implementazione del sistema proposto nominato *JPEG AI Verification Model*. Sono messi a disposizione due encoder: `Enc0`, il più semplice, implementabile su dispositivi mobili, ed `Enc1`, più complesso, composto da *attention blocks* e *transformer*, che richiedono una capacità di computazione elevata.
Sono supportati tre diversi "operation point" *SOP, BOP, HOP*, forniti rispettivamente per dispositivi dotati di CPU, dispositivi mobili, e dispositivi dotati di hardware specializzato come le GPU.
Interessante è la feature chiamata *Multi-branch Decoding* 2.1, per la quale sono presenti tre diverse *synthesys transform* 1.2.3 per ogni *operation point*: il bitstream può essere decodificato da una qualsiasi di queste tre, lasciando così la libertà al dispositivo di scegliere quale livello di complessità è più adatto per le proprie capacità computazional; anche un'immagine compressa con l'encoder più complesso, può essere decodificata da un decoder più semplice.

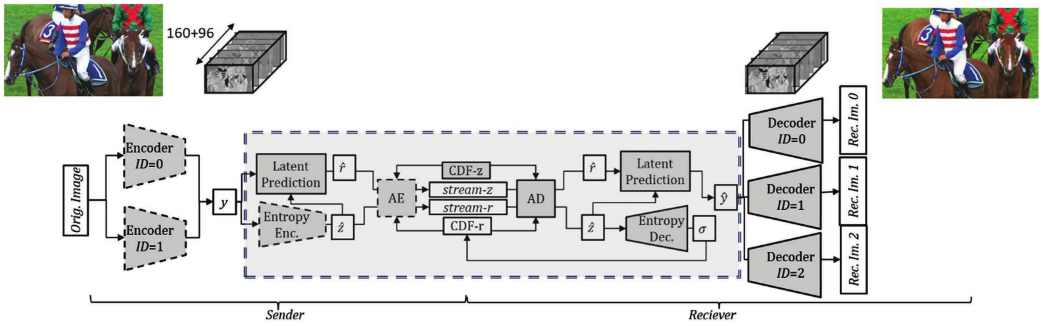


Figura 2.1: Multi-branch decoding

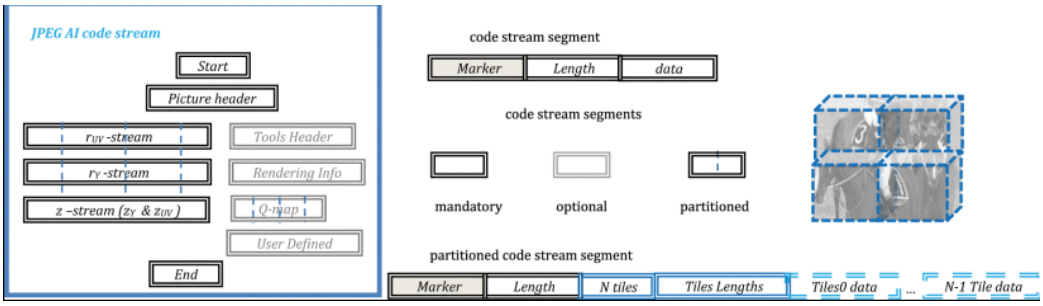


Figura 2.2: JPEG AI bitstream

Rappresentazioni estratte Le immagini prima di essere elaborate dall'encoder vengono pre-processate: inizialmente convertite nel formato YUV BT.709, separando le componenti di luminanza e crominanza, e successivamente viene applicato quello che è chiamato *Conditional Color Separation*, un approccio per cui il componente primario, la luminanza (Y), viene compresso con una rete neurale più potente, mentre la crominanza viene compressa usando informazioni della luminanza [18]; così si abbassa il picco dell'utilizzo della memoria, permettendo una riduzione nella complessità computazionale .

La separazione delle componenti è mantenuta anche nel bitstream finale 2.2, dando la possibilità di scartare la parte di crominanza se non necessaria per alcune applicazioni.

2.2 Random Forest e GridSearchCV

Come framework per la parte di machine learning è stata usata *Scikit-learn*, una libreria open-source scritta in Python, considerata come una delle più importanti per il *machine-learning*.
Come classificatori è stato scelto RandomForest, ...
spiegazione teorica di RandomForest e GridSearch???

2.2.1 Addestramento

I modelli sono stati addestrati direttamente sulle rappresentazioni latenti prodotti dall'encoder di JPEG AI, in particolare alcuni su y e altri su $\hat{y}$ (vedi fig. 1.4) per avere un confronto tra i due

- y rappresenta l'output dell'*Analysis transform* (vedi fig.1.5)
- $\hat{y}$ rappresenta l'input dell'*Synthesis transform* (vedi fig. 1.6)

L'encoder genera dei tensori separati per la luminanza e la crominanza, rispettivamente con 160 e 96 canali. *parlare del numero di immagini usate*

2.3 Dataset utilizzato

Il dataset usato negli esperimenti è *140k Real and Fake Faces* [19], una raccolta di 140.000 immagini di volti, suddivise in 70.000 immagini reali e 70.000 immagini fake.
Le immagini reali fanno parte del dataset *Flickr-Faces-HQ* (FFHQ) [20], e consistono in fotografie di volti umani in alta qualità distinte da altri dataset per la presenza di una grande varietà di soggetti per età ed etnia, ma anche per la presenza di accessori come occhiali o cappelli [5]; queste immagini sono state collezionate da Flickr, un servizio web che offre la possibilità di pubblicare immagini ad artisti e fotografi. Le immagini fake invece fanno

parte del dataset *1 Milion Fake Faces* ,un insieme di immagini generate tramite StyleGAN (vedi sez. 1.1.2.

Suddivisione dati I dati sono già suddivisi in training set, test set, e validation set, rispettivamente composti da 100000, 20000 e 20000 immagini; ogni insieme mantiene un perfetto bilanciamento tra le due classi. Il perfetto bilanciamento lo rende ideale per l'addestramento, e la suddivisione nei vari set semplifica molto la preparazione dei dati.

Capitolo 3

Esperimenti

3.1 Specifiche Hardware

3.2 Metriche di valutazione per esperimenti

3.3 Risultati

Capitolo 4

Conclusioni

Bibliografia

[1] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection. *Information Fusion*, 64:131–148, 2020.

[2] Tharindu Fernando, Darshana Priyasad, Sridha Sridharan, Arun Ross, and Clinton Fookes. Face deepfakes–a comprehensive review. 2025.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[4] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

[6] Gregory K Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[7] Sonehara, Kawato, Miyake, and Nakane. Image data compression using a neural network model. In *International 1989 Joint Conference on Neural Networks*, pages 35–41. IEEE, 1989.

[8] GL Sicuranza, GIOVANNI Ramponi, and Stefano Marsi. Artificial neural network for image compression. *Electronics letters*, 26(7):477–479, 1990.

[9] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[10] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. 2018.

[11] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

[12] J Ascenso and P Akayzi. Report on the state-of-the-art of learning based image coding. *ISO/IEC JTC 1/SC29/WG1, Geneva, Document N83058*, 2019.

[13] João Ascenso, Elena Alshina, and Touradj Ebrahimi. The jpeg ai standard: Providing efficient human and machine visual data consumption. *Ieee Multimedia*, 30(1):100–111, 2023.

[14] Nora Hofer and Rainer Böhme. A taxonomy of miscompressions: Preparing image forensics for neural compression. In *2024 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2024.

[15] Edoardo Daniele Cannas, Sara Mandelli, Nataša Popović, Ayman Alkhateeb, Alessandro Gnutti, Paolo Bestagini, and Stefano Tubaro. Is jpeg ai going to change image forensics? *arXiv preprint arXiv:2412.03261*, 2024.

[16] Sandra Bergmann, Fabian Brand, and Christian Riess. Three forensic cues for jpeg ai images. *arXiv preprint arXiv:2504.03191*, 2025.

[17] JPEG / JPEG AI / JPEG AI Reference Software · GitLab. https://gitlab.com/wg1/jpeg-ai/jpeg-ai-reference-software, July 2025.

[18] Panqi Jia, Ahmet Burakhan Koyuncu, Georgii Gaikov, Alexander Karabutov, Elena Alshina, and André Kaup. Learning-based conditional image coder using color separation. In *2022 Picture Coding Symposium (PCS)*, pages 49–53. IEEE, 2022.

[19] 140k Real and Fake Faces. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces.

[20] NVlabs/ffhq-dataset. NVIDIA Research Projects, August 2025.