
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 difficilmente distinguibili da quelli reali. 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.
Nonostante queste tecnologie abbiano applicazioni creative ed innovative in campi come quello dell'intrattenimento, ormai è estremamente facile anche per non esperti generare dei contenuti digitali falsi molto realistici, implicando molti rischi per possibili usi illeciti.
Questo lavoro di tesi triennale si propone di esaminare l'uso 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 riesce ad estrarre dalle immagini delle rappresentazioni latenti sufficientemente ricche di informazioni da riuscire a rilevare se un'immagine sia sintetica utiluzzando un semplice classificatore.

### Struttura relazione

La relazione è strutturata come segue: nel capitolo 1 vengono presentati gli aspetti teorici necessari per capire il resto del lavoro, tra cui la creazione e generazione di immagine 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, ed infine nel capitolo 4 sono tratte le conclusioni.

# Capitolo 1

# Stato dell'arte

## 1.1 Immagini fake: rischi, generazione e rilevamento

L'aumento negli 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 di alto livello di realismo un'operazione semplice ed accessibile anche a utenti non esperti.
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 infatti nasce dalla combinazione di "deeplearning" e "fake" e vuole indicare la manipolazione di contenuti già esistenti o la generazione di nuovi; su quest'ultimi si concentrerà questo lavoro di tesi.
L'uso più comune dei deepfakes riguarda la produzione e rielaborazione di immagini rappresentanti volti umani per scopi di intrattenimento o marketing; le stesse tecnologie possono però essere utilizzate per fini illeciti come la creazione di profili falsi ma apparentemente autentici, diffusione di disinfor-

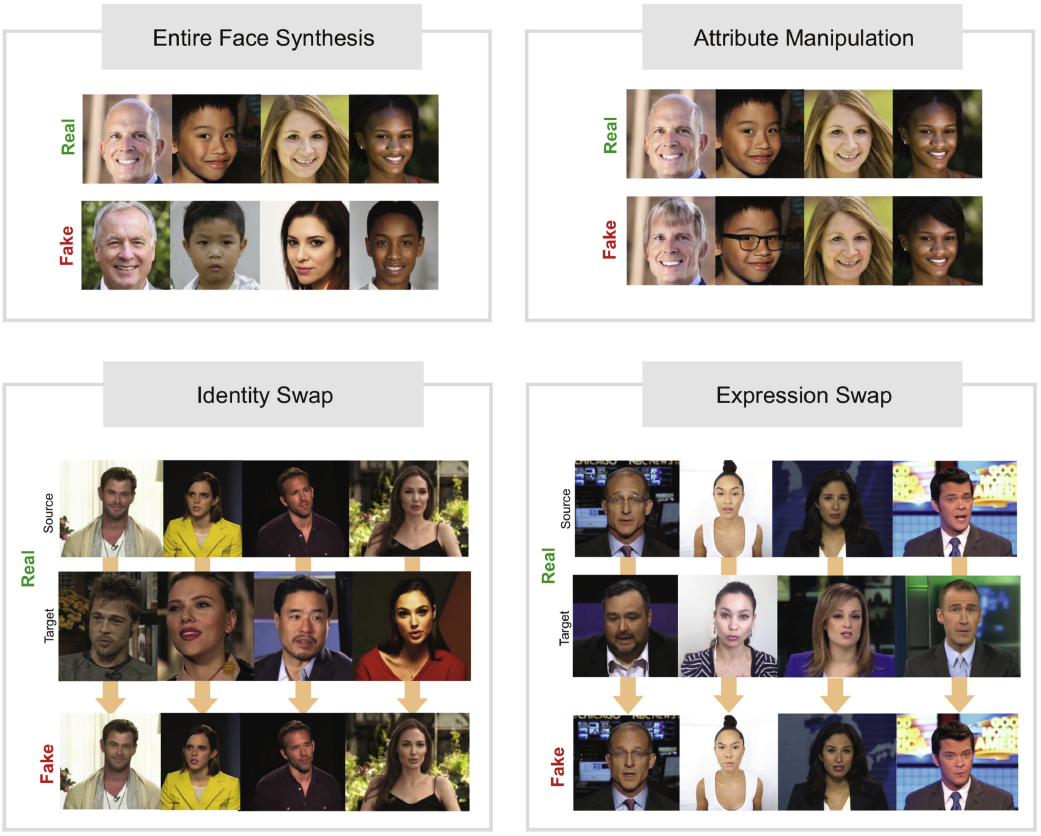


Figura 1.1: Esempi per ogni tipo di manipolazione di volti

mazione e produzione di materiale diffamatorio nei confronti di personaggi pubblici e anche 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 quando si parlerà di generazione e detection.

### 1.1.1 Generation

Per la creazione di immagini di volti sintetiche sono usate principalmente tre diverse architetture [2]: Autoencoders, GANs e Diffusion Models. Comprenderne il funzionamento è fondamentale per capire come sviluppare metodi per identificare le 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 usate nell'identity swapping e l'idea è che un encoder comune apprende una rappresentazione latente condivisa, mentre decoder separati si specializzano nella ricostruzione di immagini relative ad un individuo specifico [2] , così da ottenere uno scambio di identità decodificato la rappresentazione compatta della persona $A$ con il decoder della persona $B$.

**Diffusion Models** I modelli di diffusione sono modelli probabilistici il cui funzionamento si basa su due fasi: nel *forward process* l'immagine viene corrotta introducendo del rumore casuale, mentre nel *reverse process* il modello viene poi addestrato a ricostruire l'immagine iniziale rimuovendo il rumore. Queste fasi vengono ripetute in modo iterativo e permettono di generare immagini di alta qualità riuscendo a catturare dettagli molto fini; i Diffusion Models attualmente rappresentano lo stato dell'arte in questo campo.

### 1.1.2

GANs**??** GAN è l'architettura proposta nel 2014 da Goodfellow et al.[3] composta da due componenti principali: generatore e discriminatore. Il generatore $G$ ha il compito di produrre delle immagini realistiche partendo da un input casuale, mentre il discriminatore $D$ deve distinguere i campioni reali da quelli generati da $G$.
Le due reti vengono addestrate insieme competendo una contro l'altra in un

*minimax game* nel quale $G$ deve massimizzare la probabilità che $D$ compia un errore. Inizialmente $G$ genererà immagini rumorose e facilmente identificabili da $D$; quest'ultimo, ricevendo in input sia immagini reali che generate da $G$, fornirà in output la probabilità che un'immagine appartenga all'insieme di immagini autentiche. Il gradiente della funzione di perdita di $D$ verrà usato per aggiornare $G$, che alla fine dell'addestramento avrà imparato a generare campioni abbastanza realistici in modo da ingannare $D$.
Questa architettura è stato il primo salto di qualità nel campo della generazione di immagini e per anni è stata il punto di riferimento.

#### StyleGAN

### 1.1.3 Detection

#### Approcci classici

#### Approci basati su deeplearning

## 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, così da permettere 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 importante e 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 usata quando è richiesta una ricostruzione perfetta dell'immagine originale, per esempio nel caso di immagini mediche. Quella con perdita invece scarta parte dell'informazione, in particolare dei dettagli non percettibili dall'occhio umano, per raggiungere un maggior livello di compressione; è quella usata nella condivisione delle immagini sul web.

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

più efficiente; questi coefficienti vengono poi quantizzati, fase dove si ha 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 [4]; il suo successo è dovuto al basso costo computazionale necessario per calcolare la DCT (*Trasformata discreta del coseno*), che, applicata a blocchi di $8 \times 8$ 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; questi metodi quindi usano una trasformata fissa che ha lo svantaggio di non essere abbastanza flessibile e ottimale per tutti i tipi di immagine.

## Learned image processing

In questo nuovo paradigma di compressione si fruttano le reti neurali: si possono trovare articoli che propongono delle tecniche di questo tipo già negli anni '90 [5, 6], 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 con i codec tradizionali come JPEG è che in questo nuovo approccio ogni componente classica (trasformata, quantizzazione, codifica dell'entropia) è sostituita da una rete neurale, portando così alla creazione

di un *learned image codec*, ovvero reti neurali che apprendono trasformate *non-lineari* permettendo una rappresentazione più adatta.

**Architettura** La maggior parte delle tecniche sono basate sugli autoencoder [7], un tipo speciale di rete neurale in grado di imparare la mappatura tra l'input e uno spazio di una rappresentazione latente, spesso chiamato *bottleneck*.
L'architettura che ha riscosso più successo è quella proposta da [8] che ha dato una prova dell'efficienza di questa tecnica, migliorata successivamente da [9].
Uno degli aspetti fondamentali di queste architetture è l'approccio end-to-end nel quale tutti i moduli, dall'encoder al decoder, vengono addestrati insieme con una *global loss-function* permettendo un'ottimizzazione "unificata" che massimizza l'efficienza di compressione. Questo rappresenta un grosso vantaggio rispetto ai metodi tradizionali dove invece ogni componente veniva ottimizzata separatamente.
Si possono trovare però anche altri tipi di architettura, e come proposto in [10], si possono classificare in:

- tipo di rete neurale utilizzata: oltre a quelli già citati, sono usate anche RNN, per elaborare l'immagine in modo sequenziale, oppure GAN, che consente una miglior qualità visiva dell'immagine ricostruita.

- dimensione dell'unità di codifica: alcune tecniche scelgono di elaborare l'intera immagine, altre decidono di farlo a blocchi di piexl, riducendo la complessità

- strategia del controllo del bit rate: alcuni usano uno stesso modello per comprimere a più bitrate, altri invece usano diversi modelli ottimizzati per un singolo livello di qualità

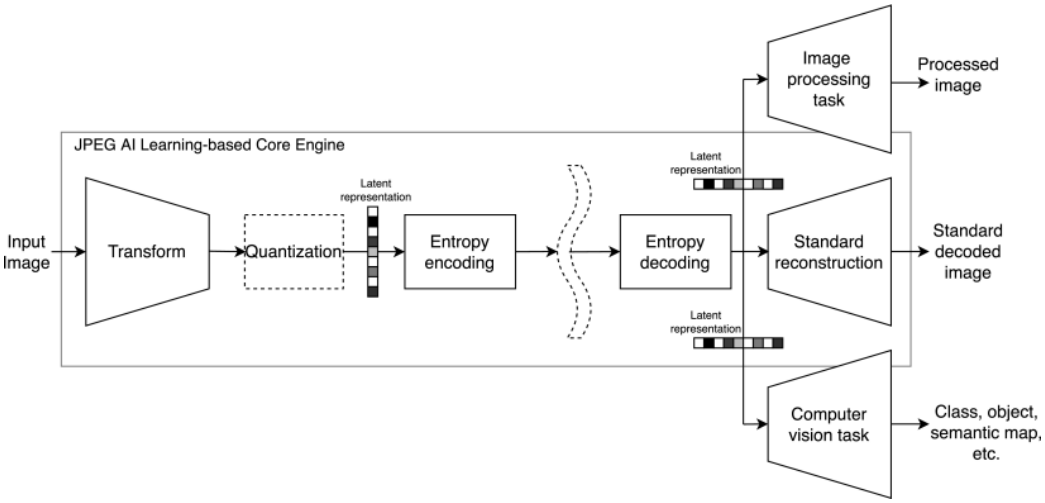


Figura 1.2: Framework JPEG AI

### 1.2.2 Motivazioni per lo sviluppo di JPEG AI

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

> "*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 effectiveperformance for image processing and computer vision tasks, with the goal of supporting royalty-free baseline* [11]"

L'obbiettivo può essere paragonato alla creazione di un *"linguaggio comune"* che consenta una rappresentazione efficiente sia per la visione degli umani che delle macchine. Questa scelta risulta interessante considerando che i contenuti digitali non sono più visualizzati esclusivamente dagli umani, ma ci sono molte applicazioni dove sono le macchine i destinatari principali. La stessa rappresentazione ricca di informazione potrebbe infatti essere usata sia per task di *image processing*, che *computer vision*, dove l'obbiettivo è estrarre dall'immagine informazione semantica, oppure potrebbe essere ricostruita (vedi fig. 1.2).

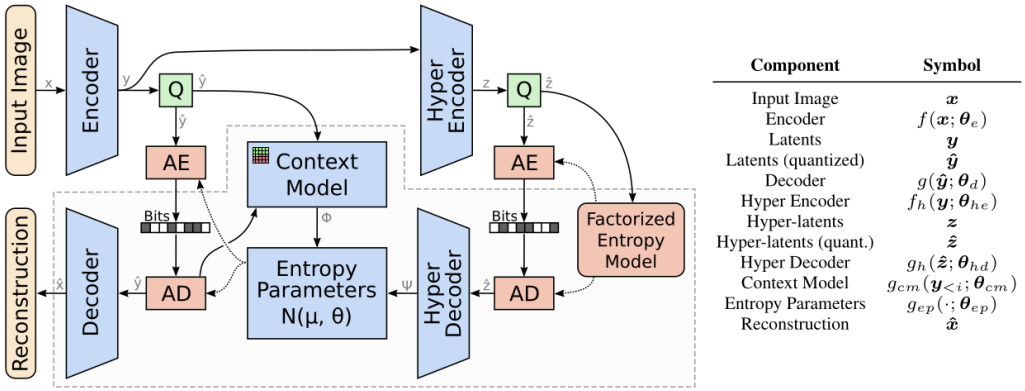


Figura 1.3: Architettura descritta in [9]

**Vantaggi** Un unico bitstream multitask offre due vantaggi: il primo riguarda la riduzione della complessità necessaria a svolgere task di image processing o computer vision, dato che consente di saltare completamente la parte di ricostruzione dell'immagine e di agire direttamente sulla rappresentazione latente; il secondo vantaggio riguarda l'accuratezza di alcune operazioni, infatti essendo stato progettato per contenere tutta l'informazione (anche semantica) contenuta nell'immagine originale, utilizzare direttamente la rappresentazione compatta garantirebbe un'accuratezza migliore di utilizzare l'immagine lossy ricostruita, in particolare per bitrate bassi.
Per ottenere questo l'encoder di JPEG AI deve generare un bitstream indipendente da tutti task, ovvero non ottimizzato per nessun compito specifico, con il requisito di mantenere sempre un alto livello di compressione.

### 1.2.3 Architettura

L'architettura di JPEG AI segue le strutture tipiche di tutti i learning-based codecs, [9] (vedi 1.3), e si può dividere in due moduli. Il modulo principale è quello che rappresenta l'autoencoder, composto dalla coppia *encoder-decoder*. L'encoder apprende una trasformata chiamata *Analysy transform*,

che mappa l'immagine $x$ in un tensore latente $y$, mentre il decoder esegue il processo inverso, *Synthesis transform*, prendendo la versione quantizzata della rappresentazione e trasformandola in immagine.
Il secondo modulo corrisponde al *Modello per l'entropia*; composto da *Hyperprior*, e *Context-model* e apprende un modello probabilistico dei latenti per svolgere in modo più efficiente la codifica dell'entropia [8]. L'*Hyperprior* è composto da un *hyper-encoder*, che estrae da $y$ delle *side-information* $z$ che vengono quantizzate ($\hat{z}$) ed incluse nel bitstream, e un *hyper-decoder*, che prendendo $z$ aiuta a predire la distribuzione dei latenti. Il *Context model* invece usa i latenti per stimare l'entropia di $\hat{y}_i$ usando gli elemeni già codificati $\hat{y}_{<i}$.
Alcuni approfondimenti sono stati fatti nella sezione 2.1
La pipeline di compressione è descritta nei seguenti paragrafi

**Fase di Encoding** Il processo di encoding di un'immagine si può osservare in 1.4: inizialmente l'analysis transform, una rete neurale composta da layer non lineari, trasforma l'immagine in un tensore latente $y$; questo viene elaborato dall'hyper-encoder che estrae da $y$ il tensore $z$, poi viene 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 genera le varianze necessarie per fare la codifica entropica dei residui, mentre un hyper-mean decoder per produrre una stima dell'immagine originale che viene usata nel modulo di Latent Prediction, che ha il compito di stimare $\mu_y$. Sottraendo $\mu_y$ 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. 1.5) avviene esattamente il processo inverso: inizialmente viene fatta la decodifica delle informazioni ausiliari $\hat{z}$ , usate per decodificare in modo più efficiente i residui, ottenndo $\hat{r}$. Attraverso un modo di latent prediciont vengono nuovamente stimati $\hat{y}$,

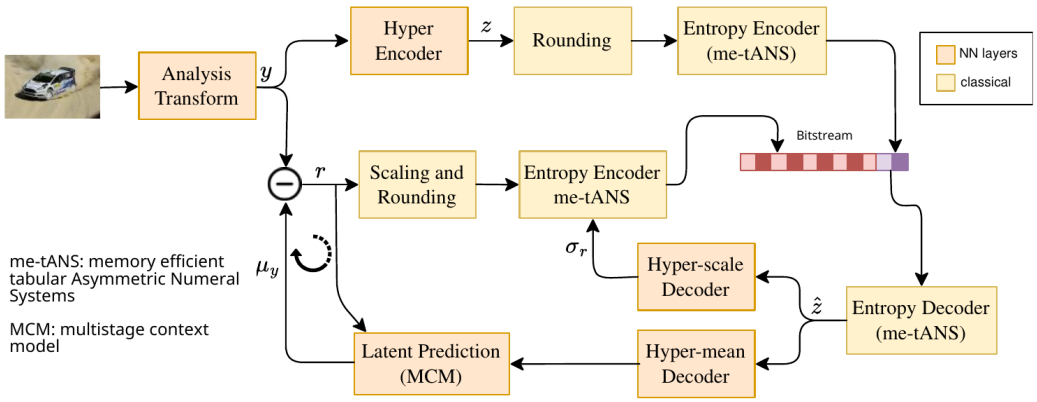


Figura 1.4: Schema della fase di encoding

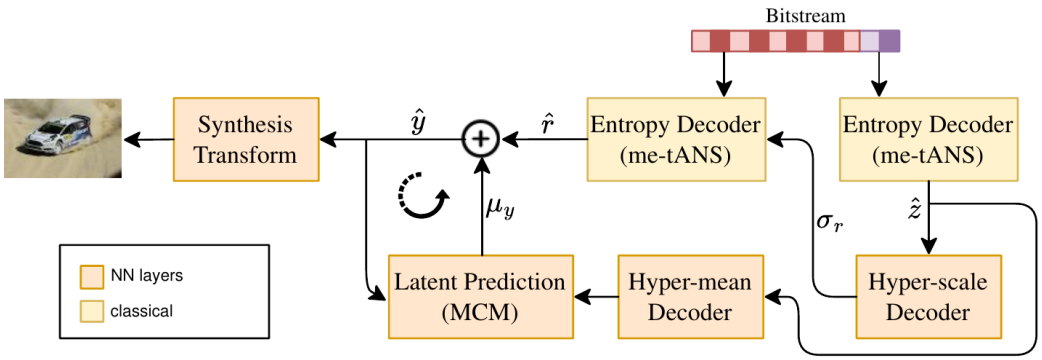


Figura 1.5: Fase di Decoding di JPEG AI

che è ricevuta dalla synthesis transform che produce un'immagine partendo dalla rappresentazione latente ricevuta.
Si rimanda alla sez. 2.1 per la descrizione della vera implementazione,

**Risultati ottenuti da JPEG AI** I primi esperimenti hanno avuto già risultati promettenti, infatti con le prime versioni delle implementazioni, JPEG AI è riuscito a raggiungere un BD-rate gain (parametro standard per la valutazione della compressione) del 28% [11] rispetto a VVC Intra (lo standard attuale più efficiente) usando la configurazione più semplice "*tools-off*", ovvero usando solo gli strumenti di base per codifica e decodifica. Inoltre è stato notato anche un miglioramento rispetto al livello di qualità soggettiva

a parità di bitrate [11], constatando l'importanza e il potenziale di questo strumento.

## 1.3 Riepilogo

In letteratura si possono già trovare molte tecniche di Learned image compression [10], ma l'importanza di JPEG AI sta nel fatto di essere diventato il primo standard ufficiale di questo tipo [12]; considerando gli obbiettivi, i requisiti, le key task dichiarate ed i risultati ottenuti 1.2.3, è possibile che questo standard verrà adottato su larga scala, rendendolo un riferimento per studi e applicazioni in futuro.
L'obbiettivo di questa tesi è capire se la rappresentazione latente prodotta dell'encoder di JPEG AI è sufficiente a task di fake detection, testando così le dichiarazioni fatte per la creazione di questo strumento.

# Capitolo 2

# Framework

## 2.1 JPEG AI Verification Model

Lo strumento utilizzato per l'estrazione delle feature è *jpeg-ai-reference-software* [13]; 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.

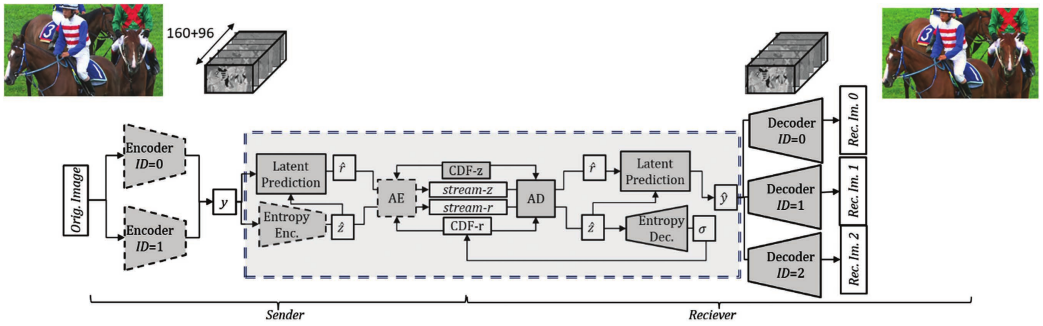


Figura 2.1: Multi-branch decoding

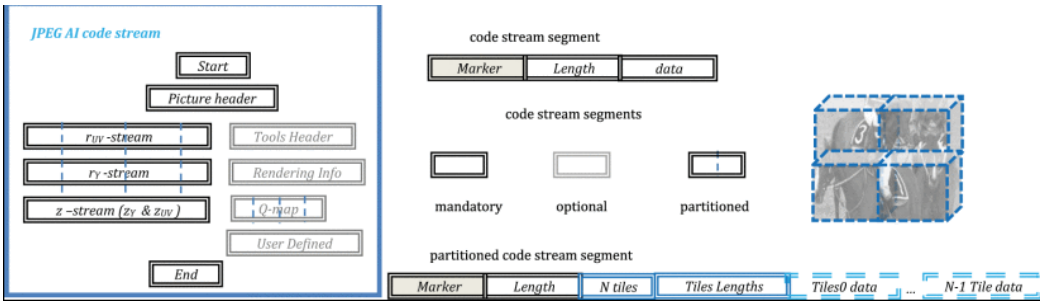


Figura 2.2: JPEG AI bitstream

**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 [14]; 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. 1.3) per avere un confronto tra i due

- $y$ rappresenta l'output dell'*Analysis transform* (vedi fig.1.4)
- $\hat{y}$ rappresenta l'input dell'*Synthesis transform* (vedi fig. 1.5)

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* [15], 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) [16], 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 [17]; 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. **??**.

**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] 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.

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

[5] 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.

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

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

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

[9] 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.

[10] 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.

[11] 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.

[12] JPEG - 106th Meeting – Online - JPEG AI becomes an International Standard.

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

[14] 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.

[15] 140k Real and Fake Faces. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces.

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

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.