

Progetto
Multimedia
Cross Domain Steganography Detection

Rosario Forte 1000001031
Giovanni Imbesi 1000006253
Docenti: Prof. Dario Allegra, Prof. Filippo Stanco

4 dicembre 2023

Indice

1 Introduzione	3
1.1 Problema	3
1.1.1 Motivazioni	3
1.1.2 Challenge	4
1.1.3 Struttura Dell'Elaborato	5
2 Prerequisiti	6
2.1 Hardware	6
2.2 Python	6
2.2.1 CUDA Compute Unified Device Architecture	6
2.2.2 PyTorch	7
2.2.3 Conda	7
2.3 Reti Neurali Convoluzionali	7
2.4 Metriche di valutazione	9
2.4.1 Accuracy	10
2.4.2 Precision	10
2.4.3 Recall	10
2.4.4 F1-score	11
2.5 Steganografia	11
2.5.1 Immagini	12
2.6 Compressione JPEG	13
2.6.1 Suddivisione in Blocchi	13
2.6.2 Trasformata discreta del coseno	13
2.6.3 Trasformata Discreta del Coseno (DCT)	13
2.6.4 Quantizzazione	14
2.6.5 Codifica Huffman	14
2.6.6 Livello di quantizzazione	14
3 Algoritmi	16
3.1 JMPOD	16
3.1.1 Distribuzione Cover	16

3.1.2	Distribuzione Stego	18
3.1.3	Detector Onnisciente	18
3.2	UNIWARD	20
3.2.1	Banchi di Filtri Direzionali	20
3.2.2	Funzione di Distorsione Spaziale	21
3.2.3	Funzione di Distorsione JPEG	21
3.2.4	Modalità D'Utilizzo	22
3.3	UERD	23
3.3.1	Original Uniform Embedding	23
3.3.2	Coefficiente di Variazione	25
3.3.3	Utilizzo dei coefficienti zero AC e DC	25
3.3.4	Funzione di Distorsione	26
3.4	Adaptive LSB	29
3.4.1	Tecnica di Sostituzione LSB	29
3.4.2	Advanced Encryption Standard	30
3.4.3	Pixel Locator Sequence (PLS)	31
3.4.4	Codifica e decodifica LSB	31
3.4.4.1	Encoding	32
3.4.4.2	Decoding	32
4	Modelli	33
4.1	EfficientNet	33
4.1.1	Scaling dei Coefficienti	34
4.1.2	Struttura del Modello	34
4.1.3	Metodo di Regolarizzazione	35
4.2	Reti Residuali (ResNet)	36
4.2.1	Innovazione dei Residual Block	36
4.2.2	Architettura delle Reti Residuali	37
4.2.3	ResNet-18	37
4.2.4	ResNet-50	37
5	Dataset	38
6	Esperimenti	39
6.1	Approccio	39
6.2	Analisi	40
6.3	Test Spaziale	41
Bibliografia		43

Capitolo 1

Introduzione

Nel seguente elaborato sarà trattato il problema di detection di steganografia, all'interno di immagini digitali, mediante l'uso di reti neurali. Verranno trattate le tematiche relative al problema, le motivazioni e le difficoltà di un tale approccio. Tutti i prerequisiti necessari nonché gli algoritmi di steganografia e di deep learning utilizzati verranno descritti nel dettaglio. In fine si propongono i benchmark ottenuti tramite i diversi esperimenti.

1.1 Problema

Nascondere un messaggio all'interno di qualsiasi forma di comunicazione è sempre stato un argomento di grande interesse per il genere umano. Nel corso degli anni, lo studio della steganografia ha portato all'evoluzione delle tecniche di occultamento dei dati, culminando nelle moderne strategie che impiegano immagini digitali come contenitori per i messaggi.

In un contesto simile, sorge la sfida di rilevare se un'immagine è stata sottoposta a steganografia mediante l'impiego di reti neurali. Il problema proposto può essere definito come steganalisi zero-shot. L'obiettivo è sviluppare un rilevatore in grado di identificare eventuali incongruenze in un'immagine e, di conseguenza, di individuare la presenza di steganografia indipendentemente dall'algoritmo utilizzato.

1.1.1 Motivazioni

La creazione di un detector universale di steganografia risponde a una serie di esigenze nell'ambito della sicurezza e della difesa digitale. Un obiettivo primario è la riduzione di attività fraudolente e cyber crimini, promuovendo la sicurezza delle comunicazioni digitali. L'instaurazione di canali di comuni-

cazione privi di informazioni nascoste costituirebbe un vantaggio strategico, soprattutto nel contesto militare e nella difesa nazionale. Tuttavia, l'importanza di un detector universale va oltre la sfera militare, estendendosi alla protezione della privacy individuale. Garantire che le comunicazioni digitali siano esenti da informazioni occulte è essenziale per preservare la privacy di individui e organizzazioni.. Nella ricerca forense digitale, l'applicazione di un detector universale semplificherebbe le indagini, consentendo agli investigatori di individuare eventuali messaggi nascosti nei documenti digitali. La sua implementazione potrebbe anche favorire la conformità alle normative di settore, garantendo che le pratiche aziendali rispettino le regolamentazioni vigenti. In ultima analisi, la ricerca e lo sviluppo tecnologico per un detector universale rappresentano un impegno avanzato nella comprensione e nella difesa contro le mutevoli tecniche di steganografia.

1.1.2 Challenge

La realizzazione di un detector universale di steganografia deve confrontarsi con sfide significative, dovute alla vasta gamma di algoritmi steganografici esistenti e alla molteplicità di domini nei quali i dati possono essere occultati. La complessità intrinseca di questo compito emerge dalla diversità di approcci adottati dagli algoritmi steganografici, che spaziano dall'incorporazione nei bit meno significativi delle immagini, all'incorporazione di dati in domini diversi da quello spaziale. L'infinita varietà di tecniche rende difficile prevedere con precisione tutti i possibili scenari in cui l'informazione potrebbe essere nascosta.

La fattibilità di un progetto di questa portata è supportata dall'osservazione che, anche se sottile e quasi impercettibile all'occhio umano, ogni immagine sottoposta a steganografia subisce una perturbazione. Questa perturbazione, anche quando minima, costituisce un segnale che può essere rilevato. È proprio questa minima alterazione che motiva la ricerca di un detector "universale", capace di discernere la presenza di incoerenze anche nelle situazioni più sottili.

La comprensione di questa piccola deviazione costituisce la chiave per lo sviluppo di un detector capace di valutare la presenza di informazioni nascoste in modo ampio e flessibile. La sfida sta nel riconoscere queste deviazioni, e nel creare un sistema in grado di adattarsi a un panorama così diversificato di possibili tecniche di occultamento dati.

1.1.3 Struttura Dell'Elaborato

Nel Capitolo 2 sono trattati tutti i prerequisiti necessari a comprendere le trattazioni seguenti. Sono inoltre forniti i setting hardware e software necessari a riprodurre gli esperimenti proposti. Nel Capitolo 3 sono descritti nel dettaglio gli algoritmi di steganografia esplorati. Il Capitolo 4 tratta dei modelli che sono stati allenati al fine di risolvere il task, fornendone i dettagli di funzionamento. Il dataset utilizzato, con le espansioni apportate e lo splitting proposto, sono descritti nel Capitolo 5. In fine i risultati ottenuti sono esplicati nel Capitolo 6.

Capitolo 2

Prerequisiti

Nel seguente capitolo saranno descritti i setup hardware utilizzati, con annessi software e librerie utili alla risoluzione del task proposto, i linguaggi di programmazione con il quale sono implementate le soluzioni proposte e i rudimenti necessari per comprendere i capitoli successivi.

2.1 Hardware

L'intero progetto è stato sviluppato su una macchina server Ubuntu, l'hardware presente sul dispositivo è il seguente:

- 2 CPU Intel(R) Xeon(R) E5-2620 v3 @ 2.40GHz.
- 32GB Memoria RAM.
- 1 Scheda video Nvidia Tesla K80

2.2 Python

Python è il linguaggio utilizzato per l'analisi dati e l'allenamento degli algoritmi di machine learning e deep learning proposti.

Nello sviluppo è stata utilizzata la versione di python 3.9.16 poiché in essa è presente il supporto per l'utilizzo delle librerie CUDA e Pytorch, descritti in seguito.

2.2.1 CUDA Compute Unified Device Architecture

CUDA è una API di calcolo parallelo su architettura GPU, fornita da Nvidia, che consente di sfruttare al meglio le capacità hardware delle differenti GPU

sul mercato. In particolare L'ecosistema CUDA si compone di più parti, ognuna mirata all'ottimizzazione dell'esecuzione di determinati processi.

Tra i diversi componenti di CUDA è presente cuDNN (CUDA Deep Neural Network Library) un modulo che fornisce l'implementazione ottimizzata dei layer principali di una neural network. CUDA e cuDNN sono utilizzati dalle librerie per la progettazione di reti neurali, come ad esempio PyTorch e Tensorflow.

Per lo sviluppo è stata utilizzata la versione CUDA 11.4.

2.2.2 PyTorch

PyTorch è un framework per la creazione di reti neurali in python, sviluppato da Meta AI. Il framework utilizza il sistema di librerie CUDA rendendo molto efficiente l'allenamento delle GPU.

PyTorch grazie alla sua struttura, basata sul paradigma di programmazione ad oggetti, consente una prototipazione veloce, nonché il riutilizzo di layer già allenati.

È stata utilizzata la versione 1.12.0.

2.2.3 Conda

Conda è un package manager open source, esso consente la creazione e la gestione di ambienti virtuali isolati. Ogni ambiente virtuale è un'installazione di python a se, pacchetti e dipendenze vivono quindi in un ambiente chiuso. La struttura degli enviroment conda consente di evitare il clash tra dipendenze, evitando quindi che pacchetti di installazioni diverse causino problemi o conflitti tra loro.

L'utilizzo di un virtual enviroment conda consente inoltre la condivisione dei pacchetti e dei settings in maniera semplice e intuitiva, di fatti è sufficiente installare il file di configurazione fornito per creare un enviroment in cui ogni dipendenza e ogni pacchetto sono predeterminati.

2.3 Reti Neurali Convoluzionali

Le reti neurali convoluzionali sono un insieme di reti neurali specializzate nella computazione di dati detti grid-like topology, di questa categoria fanno parte dati come immagini o segnali audio. Una rete neurale convoluzionale è formata da tre tipologie principali di layer diversi:

Convolutional Layer Esso è il layer di input, nonché il layer principale di una rete convoluzionale, i suoi componenti principali sono:

- Dati in input: Devono essere necessariamente grid-like topology;
- Numero di filtri da applicare all’immagine: Sono alla base del layer convoluzionale poichè ogni filtro viene applicato all’immagine generando un diverso output. I pesi dei kernel sono appresi dalla rete neurale durante l’apprendimento, va inoltre specificata la loro dimensione (solitamente 3×3). Il numero di kernel definisce la profondità del layer convoluzionale, ovvero gli output prodotti da quest’ultimo(Vedi Figura 2.1);
- Dimensione dello stride: Rappresenta il numero di pixel di cui ogni volta va spostato il kernel prima di essere riapplicato all’immagine;
- Zero-padding: Il comportamento che deve avere il filtro sui bordi dell’immagine, dove esso potrebbe eccedere e quindi non essere applicabile a causa di mancanza di informazioni;
- Funzione di attivazione: si utilizza la Rectified Linear Unit (ReLU), ovvero una funzione che dato in input x ritorna $\max(0, x)$ facendo sì che l’elemento ritornato non sia mai negativo.

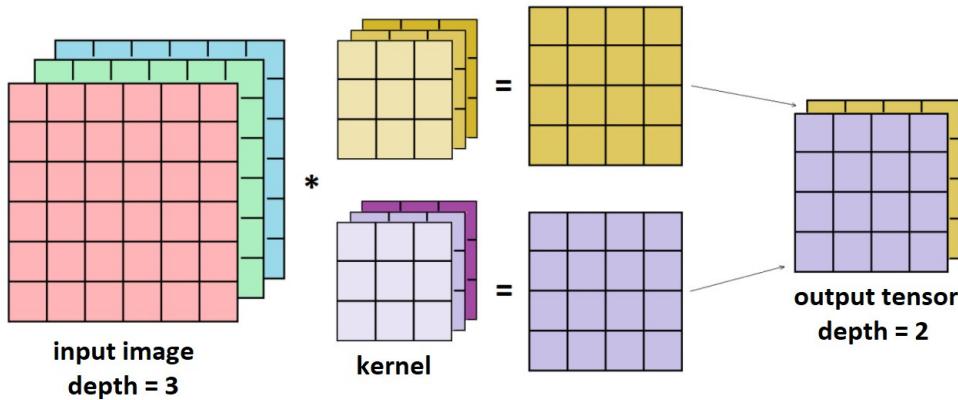


Figura 2.1: Viene mostrato come l’applicazione di due kernel produca due diverse featuremap.

Pooling Layer Questo layer è noto anche con il nome di layer di sottocampionamento, il suo comportamento è simile a quello di un layer convoluzionale, solo che in questo caso il kernel non viene appreso ma viene utilizzata una funzione di aggregazione definita a design time. Un esempio di funzione di aggregazione è il filtro di massimo.

Fully-connected Layer Si utilizza solitamente come layer di output nei task di classificazione, ogni nodo in questo layer riceve l'output di tutti i nodi del layer precedente, in questo caso viene solitamente applicata la softmax come funzione di attivazione.

2.4 Metriche di valutazione

Nella valutazione delle performance dei modelli, sono state utilizzate diverse metriche per misurare la qualità delle predizioni. Di seguito vengono introdotti i concetti di base di True Positives, True Negatives, False Positives e False Negatives:

- **True Positives (TP)**: rappresentano i valori positivi correttamente predetti dal modello. In altre parole, la ground truth è 1 e la predizione del modello è anch'essa 1.
- **True Negatives (TN)**: sono i valori negativi correttamente predetti dal modello. Qui, la ground truth e la predizione sono entrambe 0.
- **False Positives (FP)**: indicano il numero di predizioni negative che sono state erroneamente classificate come positive. In questo caso, la ground truth è 0 mentre la predizione del modello è 1.
- **False Negatives (FN)**: rappresentano le predizioni positive che sono state erroneamente classificate come negative. La ground truth è 1, ma la predizione del modello è 0.

Dopo aver definito queste nozioni è possibile introdurre le metriche di valutazione utilizzate.

2.4.1 Accuracy

L'accuracy è una metrica comune utilizzata per valutare la percentuale di predizioni corrette rispetto al totale delle predizioni effettuate. Viene calcolata come:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

L'accuracy fornisce una stima generale delle performance del modello, misurando la percentuale di predizioni corrette. Tuttavia, può non essere sufficiente in alcuni scenari, specialmente se i dati sono sbilanciati o se ci sono asimmetrie nella rilevanza delle classi.

2.4.2 Precision

La precision è una metrica che misura la proporzione di predizioni positive corrette rispetto al totale delle predizioni positive effettuate. Viene calcolata come:

$$\text{Precision} = \frac{TP}{TP + FP}$$

La precision si concentra sulle predizioni positive, misurando quanto sono accurate le predizioni del modello quando classifica un'istanza come positiva. Rappresenta quindi la capacità del modello di evitare false allerte.

2.4.3 Recall

La recall è una metrica che misura la proporzione di istanze positive correttamente predette rispetto al numerototale di istanze effettivamente positive. Viene calcolata come:

$$\text{Recall} = \frac{TP}{TP + FN}$$

La recall valuta la capacità del modello di individuare correttamente tutti gli esempi positivi presenti nel dataset. È particolarmente utile quando è fondamentale minimizzare i falsi negativi, cioè quando è importante identificare correttamente tutti gli esempi positivi.

2.4.4 F1-score

L’F1-score è una metrica che combina precision e recall utilizzando la media armonica. È un’ottima metrica da utilizzare quando si desidera bilanciare la precisione e la recall. Viene calcolata come:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

L’F1-score tiene conto sia della precision che della recall, fornendo un’unica misura complessiva delle performance del modello. È particolarmente utile quando le classi sono sbilanciate o quando è necessario trovare un compromesso tra la precisione e la recall.

2.5 Steganografia

La steganografia è l’arte di nascondere un messaggio all’interno di supporti empirici (immagini, audio, video, ecc.). In generale, un buon metodo steganografico dovrebbe presentare un’accettabile impercettibilità statistica e una capacità di carico utile sufficiente. Tuttavia, questi due obiettivi sono spesso in conflitto tra loro per un determinato algoritmo. I principali obiettivi della steganografia includono:

- **Impercettibilità statistica:** I messaggi nascosti non devono alterare significativamente le statistiche del media di copertura. L’idea è che l’osservatore non dovrebbe poter rilevare alcuna differenza significativa tra l’originale e il media con il messaggio nascosto.
- **Capacità di carico utile:** La steganografia deve consentire l’inserimento di un messaggio significativo all’interno del media di copertura, senza renderlo evidente.
- **Robustezza:** Il messaggio nascosto deve rimanere intatto anche dopo le trasformazioni o le elaborazioni del media di copertura, come la compressione o la riduzione del rumore.

Nella steganografia moderna, sono stati effettuati numerosi tentativi per raggiungere tali obiettivi. Tra di essi, il mantenimento di un modello di copertura scelto è risultato essere una cattiva idea, mentre l’approccio più comune ed efficace consiste nel minimizzare una distorsione di embedding definita in modo euristico per i media di copertura empirici.

2.5.1 Immagini

La steganografia nell'ambito delle immagini è ampiamente studiata. Nella steganografia basata su immagini è necessario distinguere 4 principali componenti (Figura 2.2):

- Messaggio segreto: Messaggio sotto una qualunque forma, convertito in bit e da incorporare nell'immagine.
- Immagine cover: Immagine nella quale verrà nascosto il messaggio segreto;
- Stego Key: Chiave crittografica utilizzata per cifrare il messaggio prima dell'embedding. È una misura di sicurezza ulteriore e opzionale.
- Immagine stego: Immagine nella quale è stato incorporato il messaggio segreto. Essa dovrebbe essere indistinguibile ad occhio nudo dall'immagine cover.

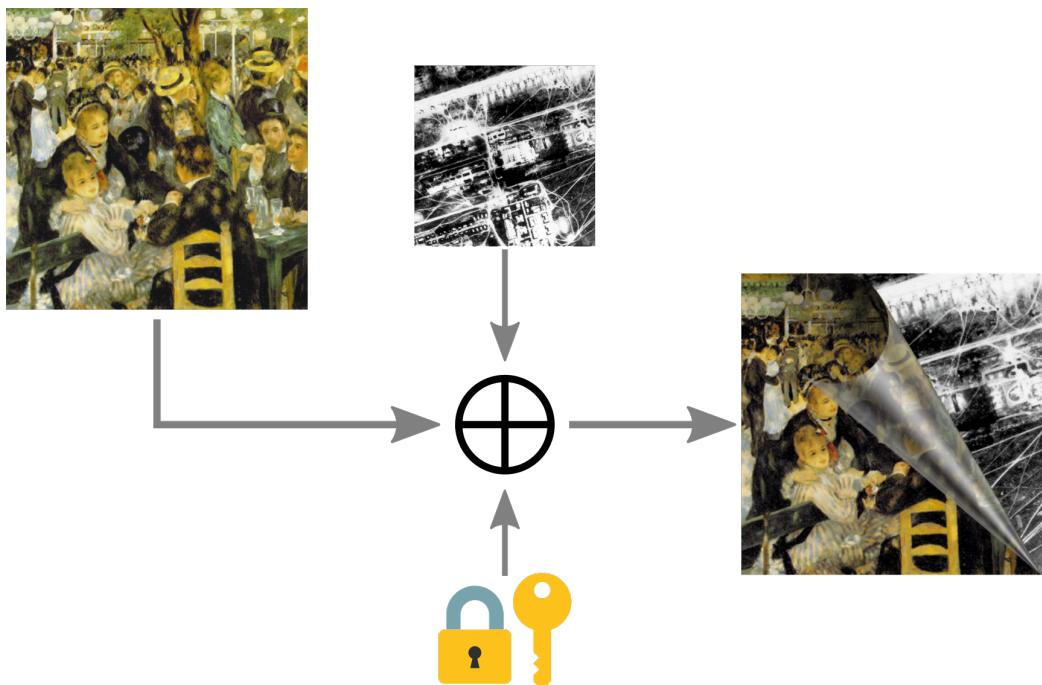


Figura 2.2: Schema concettuale di un algoritmo di steganografia.

La massima dimensione che un algoritmo di steganografia può nascondere è calcolata come il numero di bit per pixel (bpp) moltiplicato per la risoluzione dell'immagine. Nella letteratura sono solitamente utilizzate misure di bpp pari a 0.1, 0.2 o 0.4.

Un’ulteriore proprietà molto interessante delle immagini è la possibilità di rappresentarle tramite una trasformazione (DCT, FFT etc), ciò consente di nascondere dati in un dominio diverso rispetto quello di partenza, consentendo quindi la creazione di moltissimi algoritmi di steganografia con principi di funzionamento molto diversi tra loro.

2.6 Compressione JPEG

La compressione JPEG(Joint Photographic Experts Group) è una tecnica di compressione lossy concepita per ridurre le dimensioni dei file di immagini mantenendo un accettabile livello di qualità. Questo standard, sviluppato dal gruppo JPEG, rivela particolare adattabilità in scenari fotografici e in presenza di scene complesse. Di seguito ne verranno elencati le fasi principali.

2.6.1 Suddivisione in Blocchi

L’immagine viene suddivisa in blocchi quadrati non sovrapposti, di solito con dimensioni di 8x8 pixel. Questa suddivisione agevola l’applicazione efficiente della trasformata DCT, promuovendo al contempo una compressione localizzata.

2.6.2 Trasformata discreta del coseno

Il metodo di compressione JPEG si fonda sull’applicazione della trasformata discreta del coseno(DCT) per la rappresentazione dell’immagine in termini di frequenze. Tale trasformazione mira a ridurre la correlazione tra i pixel, facilitando un più efficace processo di compressione. La DCT è applicata a ciascun blocco, convertendo i dati spaziali in coefficienti di frequenza. Dato che le basse frequenze contengono informazioni visivamente più rilevanti, la DCT concentra l’energia nelle prime componenti.

2.6.3 Trasformata Discreta del Coseno (DCT)

Il fondamento della compressione JPEG risiede nell’impiego della trasformata discreta del coseno (DCT) per la rappresentazione dell’immagine in termini di frequenze. La DCT è una tecnica matematica che consente di trasformare i dati spaziali dell’immagine in coefficienti di frequenza, contribuendo significativamente alla riduzione della correlazione tra i pixel e agevolando così un processo più efficiente di compressione. Nel dettaglio, la DCT viene

applicata a ciascun blocco dell’immagine, convertendo le informazioni spaziali in una serie di coefficienti di frequenza. Questo processo è fondamentale poiché le basse frequenze contengono informazioni visivamente più rilevanti rispetto alle alte. Pertanto, la DCT opera concentrando l’energia nelle prime componenti della trasformazione.

2.6.4 Quantizzazione

Successivamente, i coefficienti DCT sono sottoposti a un processo di quantizzazione, in cui vengono arrotondati e approssimati. Tale fase introduce una perdita di informazioni, ma la quantità di tale perdita è regolata da una specifica tabella di quantizzazione.

2.6.5 Codifica Huffman

Dopo il processo di quantizzazione, che porta i coefficienti DCT ad una perdita di precisione, essi sono sottoposti a un ulteriore passo noto come codifica Huffman. La codifica Huffman opera attribuendo sequenze di bit più brevi a simboli che compaiono più frequentemente e sequenze più lunghe a quelli meno comuni. La riduzione della lunghezza dei codici per i simboli più frequenti contribuisce in modo significativo alla compressione del file, riducendo il numero complessivo di bit necessari per rappresentare l’immagine.

2.6.6 Livello di quantizzazione

La compressione JPEG richiede un delicato bilanciamento tra le dimensioni del file risultante e la qualità visiva dell’immagine compressa. Questo equilibrio è modulato principalmente attraverso il controllo di parametri chiave, con il livello di quantizzazione che assume un ruolo centrale in tale processo. Il livello di quantizzazione rappresenta un parametro determinante che incide direttamente sulla dimensione del file compresso e sulla fedeltà dell’immagine ricostruita. Livelli di quantizzazione più elevati inducono una maggiore approssimazione dei coefficienti DCT, riducendo le informazioni dettagliate e conducendo a una compressione più significativa delle dimensioni del file. Tuttavia, questa maggiore compressione si traduce spesso in una diminuzione visibile della qualità dell’immagine, con perdite di dettagli più evidenti. D’altro canto, livelli di quantizzazione più bassi preservano più dettagli dell’immagine, ma comportano una maggiore dimensione del file compresso. La scelta del livello di quantizzazione è quindi un trade-off critico tra la dimensione del file desiderata e il mantenimento della qualità visiva dell’immagine risultante.

Le fasi appena descritte sono riassunte dallo schema in Figura 2.3

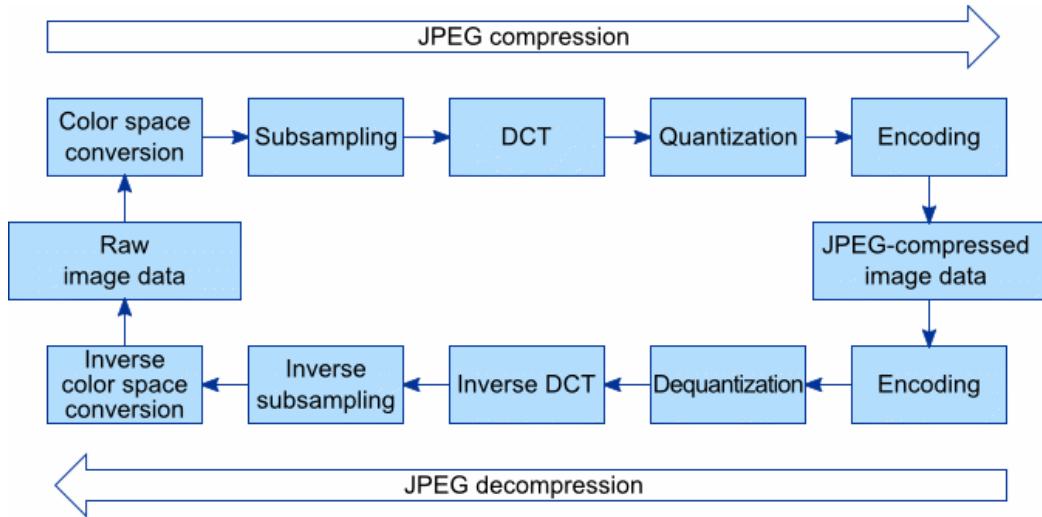


Figura 2.3: Passi principali della codifica e decodifica JPEG.

Capitolo 3

Algoritmi

Nella seguente sezione saranno illustrati i diversi algoritmi di steganografia utilizzati.

3.1 JMPOD

L'algoritmo di steganografia JMPOD [1] si basa sul trovare il miglior modo di nascondere dati all'interno dei coefficienti della DCT al fine di minimizzare le prestazioni di un detector avversario. Per fare ciò, si opera sotto l'ipotesi che il detector conosca la distribuzione statistica di cover e stego. L'obiettivo principale è quindi quello di ottenere uno stego in cui la distribuzione dei coefficienti della DCT sia praticamente indistinguibile dalla distribuzione della cover. Per fare ciò in passato sono stati esplorati diversi studi su distribuzione Laplaciana, Gaussiana e di Cauchy. Il problema principale di questi algoritmi è che essi non lavorano sui singoli coefficienti, di conseguenza l'algoritmo di steganografia assume che i coefficienti della DCT siano indipendenti e identicamente distribuiti. Diversamente, JMPOD assume che i coefficienti della DCT non siano identicamente distribuiti.

Al fine di creare un embedding che sia resistente agli attacchi è necessario analizzare le distribuzioni di probabilità dei coefficienti della DCT di immagini cover e stego, descritte nelle sezioni seguenti. Viene fornito un esempio visuale di come l'algoritmo perturbi le immagini Figura 3.1.

3.1.1 Distribuzione Cover

Nel dominio spaziale ogni pixel $x_{k,l}$ è modellato come:

$$x_{k,l} \sim \mathcal{N}(\mu_{k,l}, \sigma_{k,l}^2)$$

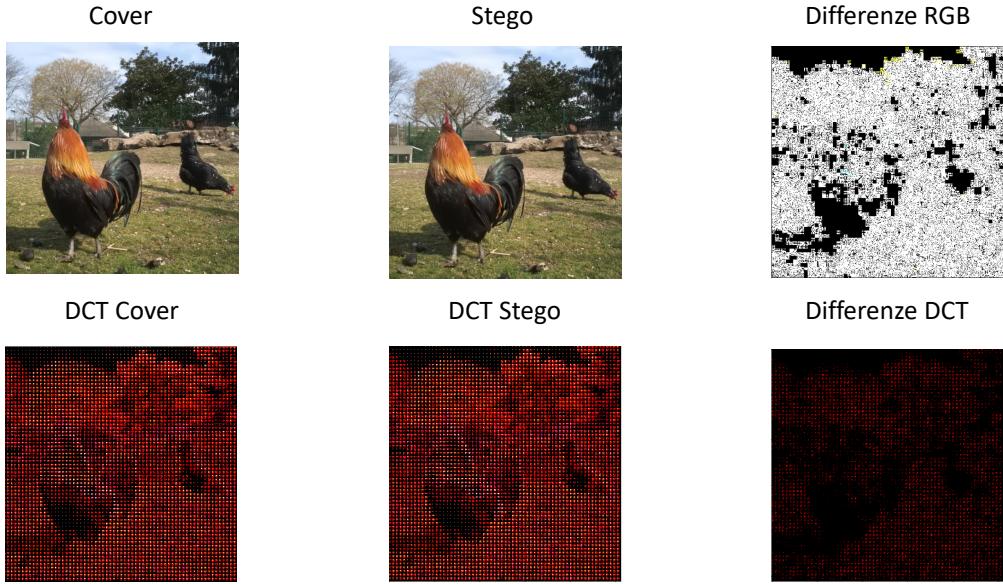


Figura 3.1: Sono mostrate le immagini cover e stego con i rispettivi coefficienti DCT. Nella colonna a destra sono evidenziate le differenze tra le immagini, altrimenti impercettibili ad occhio nudo.

dove $\mu_{k,l}, \sigma_{k,l}^2$ rappresentano rispettivamente media e deviazione standard del pixel a posizione (k,l) .

Dato \mathbf{x} il vettore rappresentante i blocchi 8×8 di pixel, i coefficienti ottenuti sono rappresentati come:

$$\mathbf{c} = \mathbf{D}\mathbf{x}$$

Dove \mathbf{D} rappresenta la trasformazione lineare della DCT.

Per i coefficienti della DCT ottenuti è effettuata l'assunzione di indipendenza:

$$\mathbf{c}_{m,n} \sim \mathcal{N}(\mu_{m,n}, \sigma_{m,n}^2)$$

Ogni coefficiente è poi quantizzato con un diverso fattore $\Delta_{m,n}$ dipendente dalla posizione.

$$\bar{c}_{m,n} = \text{round} \left(\frac{c_{m,n}}{\Delta_{m,n}} \right)$$

La seconda assunzione effettuata dall'algoritmo è che la quantizzazione effettuata è trascurabile rispetto il rumore presente nell'immagine, assunzione non sempre vera. Le assunzioni introdotte consentono di modellare la probability mass function dei coefficienti come segue:

$$p_0(k) = \mathbb{P}[\bar{c}_{m,n} = k] = \frac{\Delta_{m,n}}{\sigma_{m,n}} \phi \left(\frac{k\Delta_{m,n} - \mu_{m,n}}{\sigma_{m,n}} \right)$$

Dove ϕ rappresenta la probability density function (pdf) di una distribuzione normale.

3.1.2 Distribuzione Stego

Analizzata la distribuzione dei coefficienti delle immagini cover, è necessario modellare i coefficienti delle immagini stego. Per ottenere l'embedding nei coefficienti, quando il bit da nascondere è uguale al least significant bit (LSB) del coefficiente non viene fatto nulla, ma qualora non vi sia un match, esso può essere cambiato con un valore di ± 1 . Ogni coefficiente della DCT $c_{m,n}$ ha una probabilità di essere perturbato pari a $\beta_{m,n}$, denotato come change rate. Il massimo payload che può essere inserito è denotato come:

$$R(\beta) = \sum_{n,m} H_3(\beta_{m,n})$$

con $H_3(x)$ ternary entropy:

$$H_3(x) = -2x \log_2 x - (1 - 2x) \log_2(1 - 2x)$$

Da cui si ottiene il processo che descrive la distribuzione dei coefficienti steganografati $s_{m,n}$ come:

$$\mathbb{P}[\bar{s}_{m,n} = \bar{c}_{m,n}] = (1 - 2\beta_{m,n})$$

$$\mathbb{P}[\bar{s}_{m,n} = \bar{c}_{m,n} + 1] = \mathbb{P}[\bar{s}_{m,n} = \bar{c}_{m,n} - 1] = \beta_{m,n}$$

Dalla quale è possibile derivate la pmf dei coefficienti steganografati:

$$p_{\beta_{m,n}}(k) = (1 - 2\beta_{m,n})p_0(k) + \beta_{m,n}(p_0(k+1) + p_0(k-1))$$

3.1.3 Detector Onnisciente

Definite le distribuzioni di probabilità dei coefficienti di immagini stego e cover bisogna definire un approccio di "difesa" contro un possibile detector. Poiché non si ha nessuna informazione a priori del detector utilizzato, si suppone che esso sia onnisciente, ovvero che sia a conoscenza di $\mu_{m,n}$, $\sigma_{m,n}$ e $\beta_{m,n}$. Per chiarezza si denota con $\bar{c}_{m,n}$ i coefficienti quantizzati della DCT appartenenti all'immagine cover, $\bar{s}_{m,n}$ il campione corrispondente dopo l'inserimento del dato nascosto, e con $\bar{z}_{m,n}$ si denota un coefficiente di natura ignota (cover o stego). Il problema di trovare un dato nascosto viene definito come un test sulle ipotesi:

$$\begin{cases} \mathcal{H}_0 : \bar{z}_{m,n} \sim \mathcal{P}(\mu_{m,n}, \sigma_{m,n}; 0) \\ \mathcal{H}_1 : \bar{z}_{m,n} \sim \mathcal{Q}(\mu_{m,n}, \sigma_{m,n}; \beta_{m,n}) \end{cases}$$

Ridefinibile come la soluzione all'equazione di log likelihood ratio:

$$\log\Lambda(\mathbf{Z}) = \sum_{m,n} \log\Lambda(\bar{z}_{m,n}) = \sum_{m,n} \log \left(\frac{p_{\mathcal{H}_0}(\bar{z}_{m,n})}{p_0(\bar{z}_{m,n})} \right) \stackrel{\mathcal{H}_0}{\leq} \tau$$

La threshold τ utilizzata come decision boundary è definita come:

$$\mathbb{P}[\Lambda(\mathbf{Z}) > \tau] = \alpha_0$$

Dove α_0 è scelto in maniera tale da ottenere la significatività desiderata. Da queste informazioni, e tramite uno studio sulla funzione di log likelihood ratio, è possibile notare che essa converge asintoticamente alle seguenti distribuzioni:

$$\log\Lambda^*(\mathbf{Z}) \rightsquigarrow \begin{cases} \mathcal{N}(0, 1) & \text{soddisfatto } \mathcal{H}_0 \\ \mathcal{N}(\sqrt{2}\varrho, 1) & \text{soddisfatto } \mathcal{H}_1 \end{cases}$$

dove $\log\Lambda^*(\mathbf{Z})$ è definito come:

$$\log\Lambda^*(\mathbf{Z}) = \frac{\sum_{m,n} \log\Lambda(\bar{z}_{m,n}) - \mathbb{E}_{\mathcal{H}_0}[\log\Lambda(\bar{z}_{m,n})]}{\sqrt{\sum_{m,n} \mathbb{V}ar_{\mathcal{H}_0}[\log\Lambda(\bar{z}_{m,n})]}}$$

con $\mathbb{E}_{\mathcal{H}_0}[x]$ e $\mathbb{V}ar_{\mathcal{H}_0}[x]$, rispettivamente, valore atteso e varianza della variabile x sotto l'ipotesi \mathcal{H}_0 .

Le performance del classificatore sono totalmente dominate dal deflection coefficient, definito come:

$$\varrho = \sum_{m,n} \beta_{m,n}^2 \left(\frac{\Delta_{m,n}}{\sigma_{m,n}} \right)^4$$

Tenendo a mente l'obiettivo di nascondere un messaggio e nel frattempo minimizzare le performance del detector ideale, bisogna risolvere il seguente problema di ottimizzazione al fine di trovare il valore del deflection coefficient:

$$\beta_{m,n} = \arg \min \sum_{m,n} \beta_{m,n}^2 \left(\frac{\Delta_{m,n}}{\sigma_{m,n}} \right)^4$$

soggetto al vincolo:

$$\mathbf{R} = \sum_{m,n} \mathbf{H}_3(\beta_{m,n})$$

Risolta l'equazione e trovati tutti i $\beta_{m,n}$ è quindi possibile effettuare l'embedding dei dati come descritto in Sezione 3.1.2.

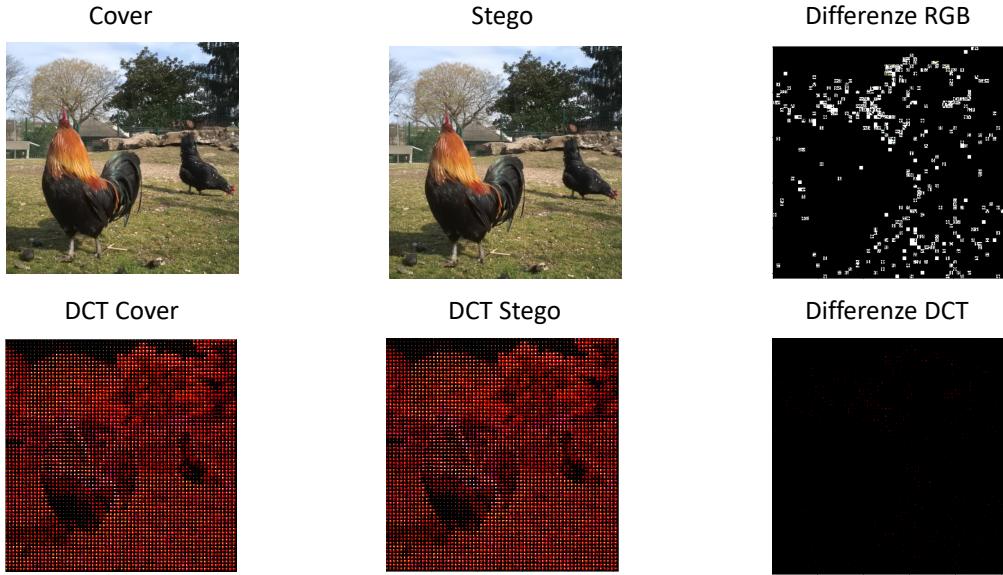


Figura 3.2: Sono mostrate le immagini cover e stego con i rispettivi coefficienti DCT. Nella colonna a destra sono evidenziate le differenze tra le immagini, altrimenti impercettibili ad occhio nudo.

3.2 UNIWARD

L’algoritmo UNIversal WAvelet Relative Distortion (UNIWARD) [2] è basato sull’uso di regioni rumorose e con alta varianza lungo le direzioni. Un’esempio di tali regioni sono gli edge o regioni textured dell’immagine. L’embedding delle informazioni viene fatto tramite il cambio di coefficienti ottenuti tramite l’applicazione di un banco di filtri direzionali. L’utilizzo di banchi di filtri direzionali ha come effetto quello di evitare le regioni uniformi dell’immagine, ciò consente di ottenere un’embedding più robusto e difficilmente rilevabile. Viene fornito un esempio visuale di come l’algoritmo perturbi le immagini Figura 3.2.

3.2.1 Banchi di Filtri Direzionali

La distorsione introdotta dal processo di embedding dipende fortemente dal banco di filtri utilizzati. Nella trattazione saranno utilizzati 3 filtri lineari e invarianti per traslazione, denotati con:

$$\mathcal{B} = \{\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \mathbf{K}^{(3)}\}$$

Essi saranno utilizzati per valutare la smoothness di un'immagine cover \mathbf{X} lungo l'asse orizzontale, verticale e obliqui tramite il calcolo dei residui direzionali ottenuti come:

$$\mathbf{W}^{(k)} = \mathbf{K}^{(k)} \star \mathbf{X}$$

dove l'operazione di \star rappresenta una convoluzione specchiata e con padding, in maniera tale che \mathbf{W} abbia la stessa dimensione di \mathbf{X} . I filtri direzionali sono creati come moltiplicazione tra filtri di decomposizione delle wavelet passa alto h e passa basso l .

$$\mathbf{K}^{(1)} = l * h^T, \mathbf{K}^{(2)} = h * l^T, \mathbf{K}^{(3)} = h * h^T$$

3.2.2 Funzione di Distorsione Spaziale

L'obiettivo è quantificare la distorsione introdotta dalla steganografia. Dati \mathbf{X} immagine cover e \mathbf{Y} immagine stego. denotiamo con $\mathbf{W}_{u,v}^k(\mathbf{X})$ e $\mathbf{W}_{u,v}^k(\mathbf{Y})$ i coefficienti delle wavelet dell'immagine cover e stego. La funzione di distorsione UNIWARD è definita come la somma dei cambiamenti dei coefficienti delle wavelet rispetto l'immagine cover:

$$\mathbf{D}(\mathbf{X}, \mathbf{Y}) = \sum_{k=1}^3 \sum_{u=1}^N \sum_{v=1}^M \frac{|\mathbf{W}_{u,v}^{(k)}(\mathbf{X}) - \mathbf{W}_{u,v}^{(k)}(\mathbf{Y})|}{|\sigma + \mathbf{W}_{u,v}^{(k)}(\mathbf{X})|}$$

Dove N ed M rappresentano la dimensione della matrice, e σ rappresenta un valore numerico necessario a stabilizzare i calcoli. Il valore di $\mathbf{D}(\mathbf{X}, \mathbf{Y})$ è minore se il cambiamento avviene in coefficienti di wavelets maggiori, ovvero zone textured o rumorose.

3.2.3 Funzione di Distorsione JPEG

Qualora si preferisca una codifica basata sui coefficienti della DCT all'interno della compressione JPEG, è possibile ridefinire la funzione di distorsione come:

$$\mathbf{D}(\mathbf{X}, \mathbf{Y}) = \mathbf{D}(J^{-1}(\mathbf{X}), J^{-1}(\mathbf{Y}))$$

dove $J^{-1}(\mathbf{X})$ e $J^{-1}(\mathbf{Y})$ rappresentano rispettivamente immagine cover e stego decompresse. Si noti che quando si opera sul dominio della DCT l'embedding acquisisce la proprietà di non additività.

3.2.4 Modalità D'Utilizzo

Al fine di utilizzare la funzione di distorsione è necessario computare il costo $\rho_{i,j}$ del cambio di un pixel, o analogamente coefficiente della DCT. Il costo del cambio di $\mathbf{X}_{i,j}$ in $\mathbf{Y}_{i,j}$ è definito come:

$$\rho(\mathbf{X}, \mathbf{Y}_{i,j}) = \mathbf{D}(\mathbf{X}, \mathbf{X}_{\sim i,j} \mathbf{Y}_{i,j})$$

dove $\mathbf{X}_{\sim i,j} \mathbf{Y}_{i,j}$ rappresenta il cambiamento nell'elemento di posizione (i, j) . Definiamo quindi il costo dell'embedding come:

$$\mathbf{D}_A(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^N \sum_{j=1}^M \rho_{i,j}(\mathbf{X}, \mathbf{Y}_{i,j}) [\mathbf{X}_{i,j} \neq \mathbf{Y}_{i,j}]$$

Dove $[S]$ è la notazione Iverson bracket. Si noti che l'utilizzo dell'embedding della UNIWARD function dà vita a diversi algoritmi sulla base del contesto di applicazione. Nel caso in cui si operi sul dominio spaziale l'algoritmo di embedding è nominato S-UNIWARD, su coefficienti DCT di una compressione JPEG è chiamato J-UNIWARD, in presenza di side information è nominato SI-UNIWARD.

3.3 UERD

L'algoritmo Uniform Embedding Revisited Distortion (UERD) [3] rappresenta un'evoluzione della strategia di uniform embedding (UED) nella steganografia JPEG. Lo scopo è quello di migliorare l'algoritmo di uniform embedding considerando come le immagini digitali possono variare statisticamente. Le modifiche apportate dall'algoritmo sono proporzionali alla variabilità naturale delle immagini. In questo modo, si cerca di rendere l'embedding più discreto e meno rilevabile, contribuendo così a migliorare l'efficacia complessiva dell'algoritmo di steganografia.

Viene fornito un esempio visuale di come l'algoritmo perturbi le immagini Figura 3.3.

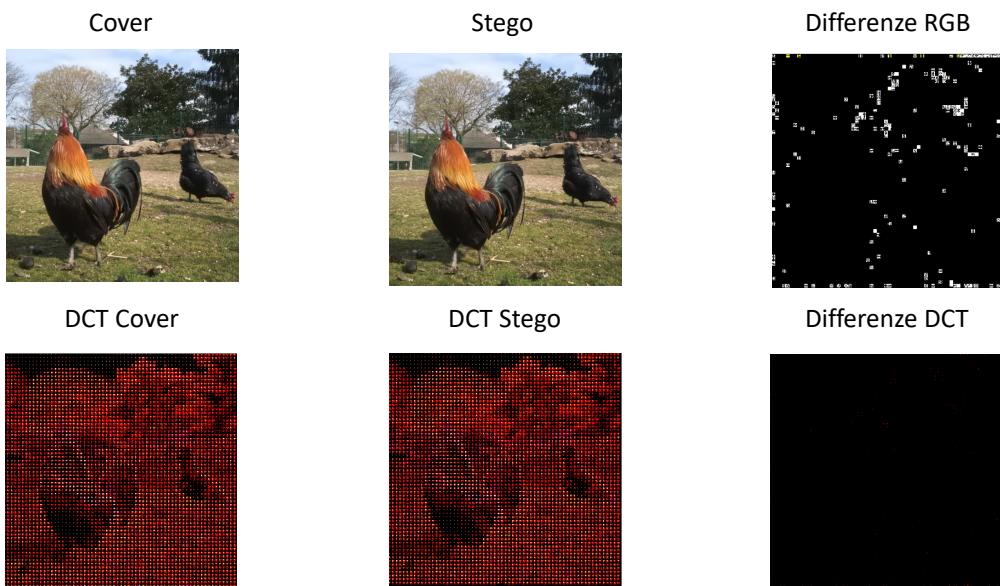


Figura 3.3: Sono mostrate le immagini cover e stego con i rispettivi coefficienti DCT. Nella colonna a destra sono evidenziate le differenze tra le immagini, altrimenti impercettibili ad occhio nudo.

3.3.1 Original Uniform Embedding

Seguendo il concetto dello "spread spectrum communication", la motivazione alla base della strategia originale di embedding uniforme è quella di "spargere" uniformemente la modifica di embedding nei coefficienti DCT quantizzati a tutte le possibili magnitudo.

In particolare, supponendo che x rappresenti i coefficienti DCT, che seguono una distribuzione laplaciana, la funzione di distorsione coinvolta in UED assume di conseguenza la forma generale della funzione $\frac{1}{|x|}$. Se dotato della corrispondente funzione di distorsione, UED comporterebbe artefatti minimi nelle statistiche del primo e del secondo ordine per l'insieme dei coefficienti DCT.

La motivazione della strategia originale di embedding uniforme può essere facilmente compresa anche dal punto di vista della teoria dell'informazione, in particolare dal principio dell'entropia massima per una sorgente discreta. Tuttavia, per le immagini naturali, le modifiche tollerabili ai coefficienti DCT con diversi valori possono variare. Le modifiche all'istogramma globale causate dall'embedding dei dati potrebbero essere trascurabili per la distribuzione di un certo coefficiente a , ma potrebbe risultare molto significativa per un coefficiente con valore b ($|a||b|$). Ciò solleva il problema se la "distribuzione uniforme" delle modifiche conduca effettivamente ad una più difficile rilevabilità nell'ambito della steganalisi.

3.3.2 Coefficiente di Variazione

Il coefficiente di variazione (CV) è una misura statistica che esprime la relazione tra la deviazione standard (σ) e la media (μ) di una distribuzione. Il CV è definito come:

$$CV = \frac{\sigma}{\mu}$$

Dove:

- - σ rappresenta la deviazione standard, che misura la variabilità o la dispersione dei dati all'interno della distribuzione.
- - μ rappresenta la media, ovvero il valore medio o tipico all'interno della distribuzione.

Il CV fornisce un'indicazione della variabilità relativa dei dati rispetto alla loro media. Più il CV è alto, maggiore è la variabilità relativa, e viceversa. In altre parole, il CV è una misura di quanto le osservazioni in una distribuzione siano disperse rispetto alla loro media. Un CV basso indica che le osservazioni sono relativamente omogenee rispetto alla media, mentre un CV alto indica che le osservazioni variano notevolmente.

Nel contesto della steganalisi, il CV viene utilizzato per valutare la variabilità relativa dei coefficienti DCT o delle matrici di co-occorrenza in base al loro valore. Maggiore è il CV, maggiore è la variabilità dei dati rispetto alla loro media, il che può influenzare la capacità di rilevamento di ciascun intervallo.

3.3.3 Utilizzo dei coefficienti zero AC e DC

Per un lungo periodo, al fine di rendere l'embedding naturalmente adattativo al contenuto, la maggior parte degli schemi steganografici JPEG evitava di ridurre il numero dei coefficienti AC zero. Con la strategia di embedding uniforme generalizzata questi costituiscono la maggior parte dei coefficienti che possono essere modificati. I coefficienti AC zero possono essere utilizzati come elementi di copertura per aumentare le prestazioni, la questione chiave è scegliere quelli appropriati per l'embedding. La scelta dei coefficienti è fatta mediante l'uso della funzione di distorsione. I coefficienti nelle regioni ad alta frequenza hanno CV relativamente bassi ed incorporare il messaggio in essi diminuirebbe anche l'efficienza della compressione JPEG, di conseguenza dovrebbero essere trattati come *wet points* nella steganografia JPEG. Se un coefficiente AC zero è un *wet point* o meno dipenderà da una funzione di distorsione unificata.

I coefficienti DC sono stati per molto tempo ignorati nell'embedding di messaggi nella codifica JPEG, in modo simile a quanto accaduto con i coefficienti AC zero. In realtà, non esiste una ragione esplicita per specificare perché i coefficienti DC non potrebbero essere utilizzati come elementi di copertura. I coefficienti DC, quando disposti insieme, possono essere considerati come un'immagine sottocampionata nel dominio spaziale. Di conseguenza, vi è solo una modesta correlazione tra un coefficiente DC e quelli adiacenti, specialmente nelle regioni textured. In altre parole, i coefficienti di variazione utilizzati per caratterizzare i coefficienti DC sono piuttosto grandi. Pertanto, è possibile incorporarvi messaggi in modo sicuro.

3.3.4 Funzione di Distorsione

In maniera analoga alla progettazione dell'UED originale, in questo caso, l'approccio consiste nell'aumentare la probabilità di modifica per i coefficienti con CV più elevati, ed allo stesso tempo, diminuire la probabilità per quelli con CV più bassi. La funzione di distorsione corrispondente, denominata UERD (distorsione dell'uniform embedding rivisitata), è stata sviluppata al fine di agevolare l'embedding a distorsione minima. La costruzione dell'UERD, prevede di analizzare i CV delle statistiche del primo e del secondo ordine per i differenti coefficienti. Dal momento che un singolo coefficiente di per sé non presenta alcuna proprietà statistica, viene investigata la "posizione" dei coefficienti, al fine di determinare quali di essi siano caratterizzati da CV più ridotti o più elevati.

Siano $x_{i,j}$ i coefficienti nella posizione (i,j) di un blocco DCT 8×8 in posizione (m,n) , la funzione di distorsione $\rho_{i,j}$ per $x_{i,j}$ è definita come:

$$\rho_{i,j} = \rho_{i,j,\text{mode}} \cdot \rho_{i,j,\text{block}}$$

dove $\rho_{i,j,\text{mode}}$ e $\rho_{i,j,\text{block}}$ sono rispettivamente le misure di distorsione per la modalità AC, e il blocco DCT corrispondente. Esistono molte scelte possibili per le due misure sopra. In tale contesto si definiscono $\rho_{i,j,\text{mode}} = q_{i,j}$, dove $q_{i,j}$ è il passo di quantizzazione corrispondente a $x_{i,j}$. $\rho_{i,j,\text{block}}$ è definito come una funzione delle energie del blocco in cui si trova $x_{i,j}$ e dei blocchi adiacenti.

Sia $x_{i,j}$ nel blocco (m,n) , la sua energia $D_{m,n}$ è definita come:

$$D_{m,n} = \sum_{k=0}^7 \sum_{l=0}^7 |x_{k,l}| \cdot q_{k,l}$$

dove $x_{k,l}$, $k,l \in \{0, \dots, 7\}$, è il coefficiente nel blocco, $x_{00} = 0$ al fine di evitare l'influenza del coefficiente DC, e $q_{k,l}$ è il passo di quantizzazione

corrispondente. Si ottiene quindi:

$$\rho_{ij} = \begin{cases} \frac{0.5 \cdot (q_{(i+1),j} + q_{i,(j+1)})}{D_{m,n} + 0.25 \sum_{d \in \hat{D}} d}, & \text{if } (i,j) \bmod 8 = (0,0) \\ \frac{q_{i,j}}{D_{m,n} + 0.25 \sum_{d \in \hat{D}} d}, & \text{otherwise,} \end{cases}$$

con $\hat{D} = \{D_{(m-1),(n-1)}, D_{(m-1),n}, D_{(m-1),(n+1)}, D_{m,(n-1)}, D_{m,(n+1)}, D_{(m+1),(n-1)}, D_{(m+1),n}, D_{(m+1),(n+1)}\}$ rappresentante le energie dei blocchi 8-vicini al blocco (m, n) .

Quando il blocco considerato è situato nel bordo dell'immagine, i blocchi inesistenti sono ottenuti tramite riempimento. Poiché le statistiche dei coefficienti DC sono significativamente diverse da quelle dei coefficienti AC, le distorsioni per i coefficienti DC sono definite in maniera euristica come la media dei coefficienti AC nel loro intorno, nello stesso blocco DCT. L'effetto dell'UERD provoca un aumento della modifica dei coefficienti di piccola magnitudo, in particolare dei coefficienti zero, e una riduzione della modifica dei coefficienti di grande magnitudine.

L'uso steganografia JPEG side-informed il principale vantaggio è l'uso dell'errore di arrotondamento durante il processo di compressione JPEG. Le side-information possono migliorare significativamente le prestazioni di sicurezza della steganografia JPEG, come dimostrato da numerosi studi. La funzione di distorsione per la steganografia JPEG side-informed può essere suddivisa in due parti, nella forma:

$$\rho = \rho_{f,s} \cdot \rho_{s,i},$$

dove $\rho_{f,s}$ è la funzione di distorsione progettata per lo spazio delle caratteristiche, mentre $\rho_{s,i}$ rappresenta la funzione di distorsione costruita con l'errore di arrotondamento. Nella pratica, $\rho_{f,s}$ può essere definita esattamente come la funzione di distorsione per la steganografia JPEG non-side-informed. Al contrario, $\rho_{s,i}$ è sempre determinata in base all'errore di arrotondamento aggiuntivo, $E = |R' - R|$, dove R rappresenta l'errore di arrotondamento dovuto alla compressione JPEG e R' è l'errore di embedding causato dall'embedding dei dati.

Pertanto, la funzione di distorsione è definita come:

$$\rho_{i,j} = e_{i,j} \cdot \begin{cases} \frac{0.5 \cdot (q_{(i+1),j} + q_{i,(j+1)})}{D_{m,n} + 0.25 \sum_{d \in \hat{D}} d}, & \text{if } (i,j) \bmod 8 = (0,0) \\ \frac{q_{i,j}}{D_{m,n} + 0.25 \sum_{d \in \hat{D}} d}, & \text{otherwise,} \end{cases},$$

dove $e_{i,j}$ rappresenta l'errore di arrotondamento aggiuntivo per $x_{i,j}$, gli altri parametri e le distorsioni dei coefficienti DC sono definiti come in precedenza.

È evidente che, per la steganografia JPEG side-informed, la minimizzazione dell'errore di arrotondamento è di particolare importanza. Come accennato in precedenza, UERD modifica molti più coefficienti con valore basso (compresi i coefficienti AC zero) rispetto a UED. Considerando il fatto che i coefficienti AC piccoli costituiscono la maggior parte di tutti i coefficienti e che gli errori di arrotondamento per tutti i coefficienti sono distribuiti in modo uniforme, UED modifica inevitabilmente meno coefficienti con un errore di arrotondamento elevato.

3.4 Adaptive LSB

Nel campo della steganografia, l'utilizzo del bit meno significativo (LSB) per nascondere dati all'interno di immagini digitali è noto da tempo. In generale, questo metodo comporta l'incorporazione sequenziale di informazioni segrete nel bit meno significativo dei pixel dell'immagine. Tuttavia, questo tipo di disposizione sequenziale può presentare una vulnerabilità, poiché un potenziale avversario, tramite l'immagine steganografata, potrebbe essere in grado di dedurre i dati nascosti, se essi non sono stati adeguatamente crittografati.

Un approccio innovativo alla steganografia basata su LSB, che si discosta dall'ordinamento sequenziale tradizionale, prevede l'uso di una sequenza unica che distribuisce la posizione dei dati nascosti in modo apparentemente casuale tra i pixel. Questa tecnica, chiamata Adaptive LSB[4], combina la dispersione dei dati con la potenza della crittografia Advanced Encryption Standard (AES). Il risultato è una sicurezza avanzata, che rende difficile per potenziali attaccanti accedere e decifrare i dati incorporati in modo sequenziale.

3.4.1 Tecnica di Sostituzione LSB

La tecnica di sostituzione LSB rappresenta uno dei metodi più comuni per la steganografia. Questa tecnica prevede che ogni bit del dato da nascondere sia sostituito con il bit meno significativo dell'immagine cover. La semplicità di implementazione ha reso il metodo basato su LSB particolarmente popolare ed efficace, in quanto l'occhio umano non è in grado di distinguere l'immagine reale dall'immagine steganografata. Questa tecnica può essere estesa a 2, 4 o persino 8 bit, ma un aumento del numero di bit utilizzati per la sostituzione può causare distorsioni e rumore nell'immagine, con conseguente perdita di qualità.

Per comprenderne il funzionamento, si consideri un'immagine come una matrice 2D di pixel. Ciascun pixel contiene un valore che dipende dal tipo e dalla profondità dell'immagine. Si prenda ad esempio la modalità RGB, in cui ciascun pixel è rappresentato da 3 componenti da 8 bit ciascuna (true color), con valori che variano da 0 a 255 (su 8 bit). È possibile convertire il messaggio in valori decimali e quindi in formato binario, utilizzando la tabella ASCII. Successivamente, per ogni pixel il bit meno significativo viene sostituito con il corrispondente bit del messaggio. In fase di decodifica del messaggio, i bit meno significativi dei pixel sono nuovamente estratti, suddivisi in gruppi di 8 e convertiti in caratteri ASCII al fine di ottenere il messaggio nascosto.

3.4.2 Advanced Encryption Standard

Advanced Encryption Standard (AES) è uno degli algoritmi simmetrici più utilizzati nel mondo della crittografia. Si tratta di una cifratura a blocchi da 128 bit con tre principali dimensioni di chiave: 128, 192 e 256 bit. AES opera su byte, piuttosto che su bit. Poiché AES ha una dimensione di blocco di 16 byte, questi byte vengono solitamente organizzati all'interno di una matrice 4x4. Ogni round in AES consiste in 4 fasi principali:

- SubBytes: Il livello SubBytes è composto da 16 S-box identici. Ciascuna S-box prende 1 byte di dati e lo trasforma applicandogli l'inverso in un campo finito, seguito da una trasformazione affine.
- ShiftRows: Ogni byte è memorizzato in una matrice 4x4 e viene trasformato in modo sistematico. Nella prima riga, non viene effettuato alcuno spostamento, nella seconda riga viene spostato di una colonna a sinistra, nella terza riga di due colonne a sinistra, e nella quarta riga di tre colonne a sinistra.
- MixColumns: In questo livello, le colonne vengono inizialmente organizzate in una matrice a colonne e moltiplicate con una speciale matrice 4x4. Questa moltiplicazione delle matrici per ciascuna colonna produce una nuova matrice 4x4 che viene quindi inviata al round successivo. Tutte le operazioni vengono eseguite in un campo finito, consentendo di distribuire le modifiche nell'intero schema di bit.
- AddRoundKey: La sottochiave generata per questo round dalla pianificazione delle chiavi viene sommata all'output del layer precedente utilizzando l'operazione XOR.

Nella sua implementazione standard, AES esegue 14 round per chiavi da 256 bit. Ogni round richiede una chiave da 128 bit univoca. AES utilizza un key-schedule per espandere una chiave breve in varie sottochiavi separate per ciascun round. Nell'ultimo round, il layer di mix column è assente. Nel processo di decriptografia, ciascun passo viene invertito prendendo l'inverso di tutte le trasformazioni nell'ordine inverso. Questa tipologia di cifratura presenta due proprietà: confusione e diffusione. Formalmente, la confusione implica che l'output finale e l'input non hanno correlazione tra loro, mentre la diffusione significa che piccoli cambiamenti nell'input devono avere un enorme effetto sull'output. In AES, la confusione è aggiunta dal layer SubBytes e la diffusione è aggiunta dai layer ShiftRows e MixColumns.

3.4.3 Pixel Locator Sequence (PLS)

La Pixel Locator Sequence (PLS) è una sequenza di pixel generata casualmente che viene creata in modo univoco per una specifica immagine. Durante il processo di decodifica, il PLS agisce come una chiave. Senza il PLS, il processo di decodifica diventa impossibile. Esso consente di introdurre casualità nel processo di codifica.

Siano N il numero totale di pixel e N_{enc} la lunghezza del messaggio codificato. Il numero di pixel N_p necessari per codificare il messaggio dato può essere calcolato attraverso la formula:

$$N_p = 3 \cdot N_{\text{enc}}$$

Ciascuno dei 3 pixel raggruppati conterrà le informazioni relative a un carattere presente nel testo cifrato. La distribuzione casuale dei pixel in PLS può essere ottenuta utilizzando la Modern Fisher-Yates Shuffle, un algoritmo che mescola casualmente gli elementi di una lista in modo che siano disposti in un ordine casuale.

3.4.4 Codifica e decodifica LSB

Al fine di migliorare la sicurezza nella steganografia basata su LSB, è stata introdotto l'utilizzo di Pixel Location Sequence (PLS) durante le fasi di codifica e decodifica. Questo approccio consente di nascondere il testo all'interno di una sequenza casuale di pixel all'interno dell'immagine, rompendo con il tradizionale metodo di iterazione sistematica dei pixel in ordine sequenziale. Fino ad ora, la steganografia basata su LSB prevedeva l'iterazione dei pixel in un ordine prevedibile. L'utilizzo di PLS consente di codificare i dati in modo casuale, aumentando significativamente il livello complessivo di sicurezza. Per ulteriori misure di sicurezza, il testo viene cifrato prima di essere incorporato nei pixel dell'immagine. Affinché la decodifica sia possibile, è necessario inviare la sequenza PLS insieme all'immagine steganografata. Pertanto, il PLS deve essere cifrato con l'algoritmo AES prima di essere trasmesso al destinatario. Una volta ricevuto, il destinatario decifra il PLS utilizzando AES. Successivamente, un algoritmo specifico percorrerà la sequenza PLS. Da ciascun pixel, verrà estratto il testo nascosto nel bit meno significativo. Il testo risultante da questa operazione sarà cifrato e richiederà ulteriori operazioni di decriptografia per ottenere il testo originale.

3.4.4.1 Encoding

La sequenza PLS creata è utilizzata per distribuire i dati tra i pixel dell'immagine. Si consideri un'immagine composta da una matrice $[N] \times [M]$ in cui N e M sono numeri interi rappresentanti il numero di pixel nelle direzioni verticale e orizzontale rispettivamente. Sia X la posizione del pixel nella PLS in cui i dati devono essere memorizzati, il valore della riga e della colonna di X può essere facilmente calcolato utilizzando

$$\text{row} = \frac{X}{M}, \quad \text{column} = X \% M$$

Ogni byte di dati cifrati viene convertito nel suo codice binario a 8 bit utilizzando i valori ASCII. I valori dei pixel vengono letti da sinistra a destra, successivamente, viene creato un gruppo di tre pixel che fornirà un totale di nove valori RGB. I valori RGB del pixel vengono resi pari o dispari a seconda della parità dei bit di dati.

3.4.4.2 Decoding

Per la decodifica, è necessario leggere sequenzialmente ogni gruppo di tre pixel da PLS. Questi generano un insieme di nove valori RGB distinti i quali vengono archiviati in una struttura dati a forma di array. Se il valore RGB è dispari, viene aggiunto un bit '1' alla stringa binaria, al contrario, se è pari, viene aggiunto un bit '0'. Successivamente, questa sequenza binaria viene convertita nel carattere corrispondente e viene unita alla stringa cifrata finale. Per decodificare il messaggio effettivo, si utilizza l'algoritmo AES per decifrare la stringa cifrata.

Capitolo 4

Modelli

Nella sezione attuale saranno presentate le reti neurali addestrate per effettuare il task di steganalisi .

4.1 EfficientNet

EfficientNet è una famiglia di modelli di reti neurali convoluzionali (CNN) progettati per ottenere elevata precisione ed efficienza nelle attività di classificazione delle immagini. Essa è stata introdotta da un team di ricercatori di Google attraverso un documento pubblicato nel 2019 dal titolo EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [5].

Gli sviluppatori di EfficientNet hanno sviluppato questi modelli basandosi sulla strategia di scaling sistematico dell’architettura delle CNN, cercando di renderle più efficienti. Questo obiettivo è stato raggiunto mediante l’utilizzo di convoluzioni separabili in profondità, che riducono il costo computazionale del modello, e un innovativo metodo di scaling che regola in maniera bilanciata la risoluzione, la profondità e la larghezza della rete. Il risultato è una famiglia di modelli che sono significativamente più precisi rispetto ai modelli precedenti e, allo stesso tempo, più efficienti dal punto di vista delle risorse computazionali. Questa sezione esplorerà in dettaglio le caratteristiche principali di EfficientNet.

4.1.1 Scaling dei Coefficienti

EfficientNet utilizza una strategia di scaling per adattare la rete a diverse dimensioni e complessità del problema. Tale strategia regola contemporaneamente tre dimensioni chiave della rete:

1. Profondità della Rete: Questa dimensione viene incrementata aggiungendo ulteriori blocchi residui, consentendo alla rete di catturare feature di crescente complessità.
2. Larghezza delle Feature Maps: Il numero di canali all'interno di ciascun blocco viene aumentato per acquisire informazioni più dettagliate.
3. Risoluzione dell'Input: La dimensione dell'input viene adattata per gestire immagini di varie dimensioni.

Questo processo di tuning delle dimensioni è attentamente bilanciato per garantire un rapporto ottimale tra prestazioni e complessità del modello.

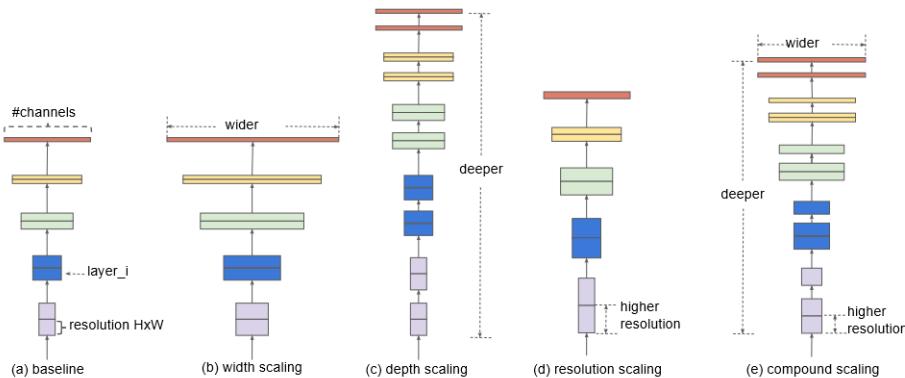


Figura 4.1: Diverse configurazioni di una MobileNet; (a) Configurazione di base di una MobileNet; (b) Mobilenet dove la dimensione delle feature apprese è stata incrementata (Scaling in larghezza); (c) MobileNet con l'aggiunta di layer (Scaling in profondità); (d) MobileNet con input ad una risoluzione più alta (Scaling in risoluzione); (e) Mobilenet incrementata tramite le diverse tipologie di scaling, bilanciate al fine di ottenerne l'accuratezza massima (Scaling compound).

Le diverse configurazioni di una MobileNet con le possibili modifiche strutturali sono riportate in Figura 4.1

4.1.2 Struttura del Modello

EfficientNet utilizza un'architettura specifica di reti neurali convoluzionali chiamata MobileNetV2 [6]. Tale architettura è composta da una serie di

convoluzioni separabili in profondità, comunemente note come una struttura a *bottleneck*. Questo tipo di architettura è noto per essere più efficiente dal punto di vista computazionale rispetto agli strati convoluzionali tradizionali, pur mantenendo un'alta accuratezza.

Nell'architettura di EfficientNet, l'immagine di input viene inizialmente sottoposta a una serie di strati convoluzionali che riducono la risoluzione dell'immagine, aumentando contemporaneamente il numero di canali. Successivamente, segue una serie di strati "bottleneck", composti da una convoluzione separabile in profondità seguita da una convoluzione puntuale. Questi strati riducono il costo computazionale complessivo del modello, incrementandone la profondità. Infine, l'output dei layer "bottleneck" passa attraverso una serie di layer completamente connessi che generano l'output finale del modello. L'output finale è un vettore di probabilità, con ogni componente associata a ciascuna classe nel dataset, indicando la probabilità che l'immagine di input appartenga a quest'ultima.

All'interno di ciascun blocco, EfficientNet adotta una strategia di "squeeze-and-excitation" per ponderare attentamente i canali delle feature maps in base alla loro rilevanza, migliorando la selettività della rete.

4.1.3 Metodo di Regolarizzazione

EfficientNet utilizza il metodo di regolarizzazione "DropConnect", simile al Dropout ma con una diversa implementazione. Il DropConnect attenua i pesi della rete durante l'addestramento, riducendo l'overfitting e aumentando la robustezza del modello. Inoltre, EfficientNet introduce un modulo di attivazione chiamato "swish," che sostituisce la tradizionale funzione di attivazione ReLU. Questo modulo "swish" migliora la convergenza del modello, consentendo di ottenere risultati più precisi. EfficientNet è ampiamente utilizzato in applicazioni di visione artificiale che richiedono un equilibrio tra efficienza computazionale e prestazioni di alto livello. La sua architettura scalabile e le tecniche di regolarizzazione avanzate lo rendono un'opzione ideale per vari contesti di deep learning.

4.2 Reti Residuali (ResNet)

Le reti neurali profonde si scontrano con il problema del vanishing gradient durante la fase di addestramento. Fenomeno per cui la backpropagation provoca una graduale diminuzione del gradiente attraverso gli strati della rete. Per superare questo problema, le reti residuali (ResNet) hanno introdotto i residual block.

4.2.1 Innovazione dei Residual Block

I residual block rappresentano una soluzione innovativa al problema del vanishing gradient. Anziché focalizzarsi esclusivamente sull'apprendimento della rappresentazione finale desiderata, un residual block si concentra sugli errori residui, cioè le discrepanze tra l'output atteso e l'output corrente del blocco (Figura 4.2).

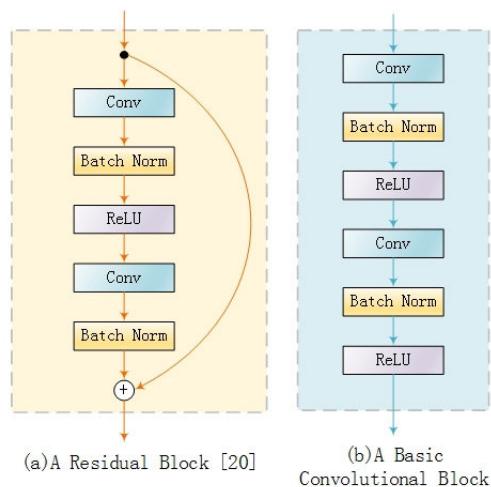


Figura 4.2: A sinistra un esempio di blocco residuale, a destra una rete convoluzionale classica.

Gli errori residui vengono sommati direttamente all'input originale, bypassando gli strati intermedi e facilitando il flusso degli errori attraverso la rete. Il collegamento residuo è formalmente espresso come segue, con $\mathcal{F}(\mathbf{x})$ come la trasformazione appresa dal blocco residuo:

$$\text{Output} = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

Dove \mathbf{x} è l'input originale al blocco residuo.

4.2.2 Architettura delle Reti Residuali

La struttura di base di una rete residuale è costituita da una successione di residual block. Ciascun blocco include layer convoluzionali, seguiti da layer di normalizzazione e attivazione, con l'aggiunta di un collegamento residuo che connette direttamente l'input all'output del blocco. Questo approccio permette la creazione di reti neurali profonde mantenendo una buona stabilità durante l'addestramento, contribuendo significativamente all'efficacia e alla capacità di apprendimento della rete.

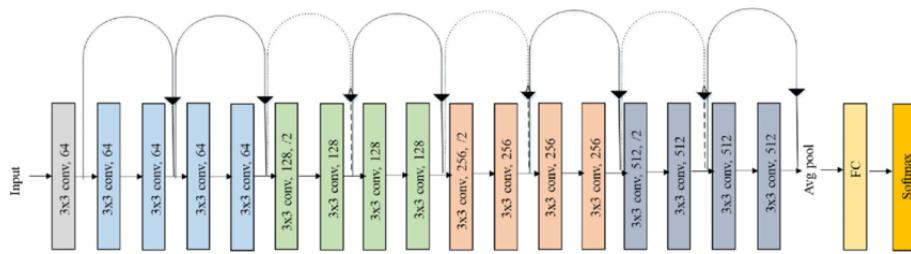


Figura 4.3: In figura un esempio dell'architettura della Resnet-18.

4.2.3 ResNet-18

ResNet-18 è una specifica implementazione di ResNet con 18 strati. Ogni blocco residuo di ResNet-18 comprende due strati convoluzionali 3×3 con funzione di attivazione ReLU. Il modello, oltre alla sua architettura più leggera, è stato progettata con un focus particolare sulla velocità di addestramento e inferenza. Grazie alla sua struttura relativamente meno profonda, ResNet-18 è spesso preferito in scenari in cui le risorse computazionali sono limitate. La sua capacità di mantenere alte prestazioni con un numero relativamente ridotto di parametri la rende una scelta popolare per applicazioni in tempo reale o con vincoli computazionali.

4.2.4 ResNet-50

ResNet-50 è una variante più profonda di ResNet con 50 strati. Rispetto a ResNet-18, ogni blocco residuo di ResNet-50 contiene tre strati convoluzionali, introducendo maggiore complessità. Questa profondità extra consente al modello di catturare rappresentazioni più intricate, rendendolo adatto a compiti più complessi.

Capitolo 5

Dataset

Al fine di allenare e valutare le reti neurali proposte è stato necessario cercare un dataset adatto alle esigenze descritte nei capitoli precedenti.

Si è scelto il dataset Alaska2 [7], proposto per la prima volta il 27 – 04 – 2020 in una challenge kaggle. Il dataset si compone di 300.000 immagini di cui 75.000 sono cover, per ognuna delle quali è fornita la relativa immagine steganografata con gli algoritmi JUNIWARD, JMPOD, e UERD. Le immagini cover sono state studiate in maniera tale da avere 3 set da 25.000 immagini con differenti quality factor, esse inoltre rappresentano scene in domini diversi, come ad esempio fattorie, animali, persone, città e via dicendo. Un’ulteriore osservazione da effettuare è che i creatori del dataset hanno cercato di rendere l’embedding il più sparso possibile attraverso i 3 canali RGB, in maniera tale da avere approssimativamente la stessa quantità di bit nascosti per ogni canale. Lo splitting utilizzato per valutare i modelli è quello proposto nella challenge.

Al fine di ottenere la valutazione cross domain, sono state predisposte 10.000 immagini di test, di cui 5.000 steganografate tramite LSB e 5.000 tramite SUNDIWARD. Ciò ha consentito di valutare la capacità di domain shift adaptation dei modelli.

Capitolo 6

Esperimenti

Nel seguente capitolo verranno descritti gli esperimenti eseguiti, le prestazioni ottenute e quali sono state le difficoltà affrontate.

6.1 Approccio

Al fine di effettuare gli esperimenti sono state addestrate diverse configurazioni di reti neurali Resnet ed EfficientNet.

In un primo step sono state allenate le diverse reti sul training set proposto nel Capitolo 5. L’addestramento è stato effettuato in primo luogo sul task di classificazione binaria, a causa però della natura sbilanciata del dataset è stata ottenuta un’accuratezza fittizia del 75%. Analizzando l’output delle reti neurali, è emerso un chiaro caso di overfitting, in quanto le predizioni mostravano una tendenza costante ad essere della sola classe positiva.

In un secondo momento si è deciso di passare ad una classificazione multiclasse, una per ogni categoria di immagine. Questo approccio obbliga la rete ad adattarsi alle diverse tipologie di dati al fine di diminuire la propria loss function. In questo caso si è riscontrata una certa significatività nelle features estratte, portando l’accuratezza della classificazione multiclasse, in fase di training, al 69%.

Per ottenere la classificazione binaria finale, si è scelto di utilizzare la rete trainata sul task multiclasse modificandone l’ultimo layer. L’addestramento è stato effettuato secondo due metodologie, la prima trainando nuovamente l’intera rete, la seconda trainando solo l’ultimo layer. Ottenendo un’accuratezza massima dell’82%.

Ogni round di training per le reti proposte utilizza circa 165 ore di GPU, per un totale di circa 2.400 ore di esperimenti con GPU in esecuzione. Di conseguenza il numero di esperimenti effettuabili e la dimensione del dataset

sono stati gli stretti necessari per ottenere delle prestazioni soddisfacenti e che quantomeno fossero analizzabili al fine di espansioni future con hardware adeguato.

6.2 Analisi

A seguito del fallimentare approccio di classificazione binaria si è passati ad un approccio multclasse. Sono state provate 2 configurazioni diverse di ResNet18, 2 configurazioni di ResNet50 e 1 configurazione di EfficientNet.

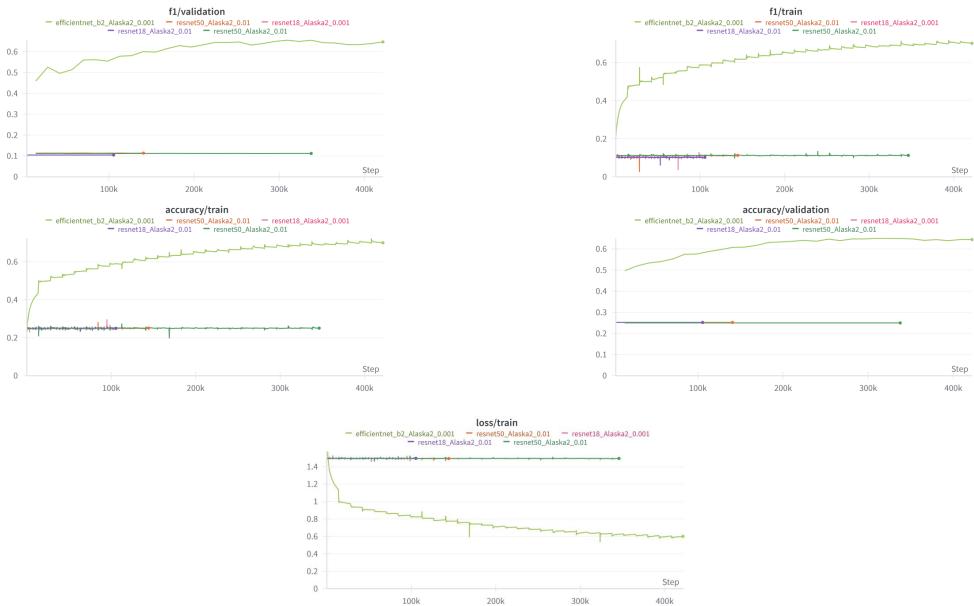


Figura 6.1: Prestazioni ottenute dalle diverse configurazioni proposte.

In Figura 6.1 è possibile notare come la ResNet non sia in grado di estrarre nessuna informazione utile dalle immagini. A causa della sua struttura, essa non è capace di estrarre informazioni che non siano "visuali", ciò limita l'archittura, confermando l'efficacia degli algoritmi di steganografia.

Le capacità dell'EfficientNet in questo caso si dimostrano nettamente superiori, com'è possibile notare dai grafici proposti. Le features apprese risultano molto informative, infatti, è stato possibile ottenere un'accuratezza dell'89% sui dati di test ALASKA.

6.3 Test Spaziale

Al fine di ottenere benchmark su un dominio diverso da quello di partenza, è stata utilizzata la configurazione migliore di EfficientNet su il test set proposto. Sui dati con steganografia LSB è stata ottenuta un'accuratezza pari a 88.78%. Nei dati crittografati tramite S-UNIWARD l'accuratezza raggiunta è pari a 87.60%. È evidente che la steganografia spaziale, risulti molto più facile da scoprire rispetto la steganografia su coefficienti DCT.

Secondo i test condotti, è molto probabile che la rete addestrata riesca a comprendere le alterazioni dei bit sulle immagini, pur non lavorando direttamente sul dominio spaziale. Si può quindi concludere che addestrare reti a lavorare sul dominio dei coefficienti della DCT possa portare alla creazione di reti capaci di individuare steganografia multi dominio.

Conclusioni

In questo elaborato è stato affrontato il problema della cross domain steganography detection. L'utilizzo di algoritmi di steganografia basati su compressione JPEG e coefficienti DCT è stato utile al fine di addestrare diverse reti neurali per il soddisfacimento del task. In una prima fase di scoperta ed esplorazione molte configurazioni sono risultate fallimentari. Successivamente tramite l'uso di una EfficientNet adeguatamente trainata, tramite diversi "trick", è stato possibile ottenere un'accuratezza soddisfacente nel test set.

Successivamente, la rete ottenuta è stata provata su algoritmi di steganografia spaziale adeguatamente adattati al task. Si è visto come la rete ottenessesse prestazioni maggiori sul nuovo dominio. Gli obiettivi preposti sono stati raggiunti, si reputa però che il task non sia concluso ma che con adeguata potenza di calcolo, e con uno studio mirato su diverse tipologie di steganografia in domini differenti si potrebbero ottenere prestazioni molto maggiori.

Bibliografia

- [1] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. Efficient steganography in jpeg images by minimizing performance of optimal detector. *IEEE Transactions on Information Forensics and Security*, 17:1328–1343, 2022.
- [2] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014:1–13, 2014.
- [3] Linjie Guo, Jiangqun Ni, Wenkang Su, Chengpei Tang, and Yun-Qing Shi. Using statistical image model for jpeg steganography: Uniform embedding revisited. *IEEE Transactions on Information Forensics and Security*, 10(12), 2015.
- [4] Sahil Gangurde and Krishnakant Tiwari. Lsb steganography using pixel locator sequence with aes. *arXiv preprint arXiv:2012.02494*, pages 1–5, 2020.
- [5] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv:1905.11946*, 2019.
- [6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.
- [7] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. Alaska#2: Challenging academic research on steganalysis with realistic images. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–5, 2020.