UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Identificazione della fotocamera sorgente da contenuti multimediali applicabile all'analisi forense

Relatore: Prof. Loris Nanni

Laureando: Enrico Napolitan

ANNO ACCADEMICO 2021 – 2022

Data di laurea 14 Novembre 2022

Sommario

Nel corso degli ultimi anni, soprattutto grazie alla digitalizzazione che il Covid19 ha portato nelle case di tutti, siamo stati protagonisti di un forte sviluppo dei cosiddetti contenuti digitali: foto e video che possono essere pubblicati in internet da chiunque lo voglia. Parallelamente, però, siamo anche stati vittime di un considerevole aumento dei crimini informatici. La polizia postale riporta che nel 2021 c'è stato un aumento di oltre il 436% nelle truffe online e addirittura 5316 casi di cyberbullismo. Dato questo angosciante aumento dei cybercrimini, illustro in questo elaborato un modo all'avanguardia per contrastarne la crescita. Infatti, sfrutto una caratteristica presente in tutte le fotocamere: il PRNU, definito da molti l'impronta digitale delle camere digitali. Quest'ultimo è una componente del rumore dell'immagine che, per fattori derivanti dalle proprietà dei componenti elettronici interni del dispositivo, è presente in ogni foto o frame di video. Il PRNU ci permette, tramite l'impiego del Deep Learning, di analizzare le immagini condivise nel web e identificare con certezza qual'è stato l'esatto dispositivo che ha scattato una determinata foto. L'obiettivo sarà quello di analizzare un lavoro prodotto da altri ricercatori correlato a quest'ambito e, successivamente, proporre un metodo alternativo per la classificazione del rumore. Nel dettaglio propongo una metodologia più consistente per la selezione delle aree dell'immagine che si presume contengano del rumore, in questo modo è possibile ridurre il quantitativo di parti analizzate di ciascun frame e, di conseguenza, è possibile impiegare più foto e più video per l'addestramento della rete neurale convoluzionale. I risultati ottenuti utilizzando come set di video il Dresden Image Database sono molto promettenti: un'accuratezza di oltre il 99% riducendo notevolmente i tempi computazionali e di spazio.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

Capitolo 1

Introduzione

1.1 Scopo

Con il continuo e sempre più importante aumento della creazione e quindi della fruizione di contenuti digitali, si sta sviluppando anche la ricerca legata a come sfruttare al meglio le nuove tecnologie, in particolare come prevenire che queste vengano impiegate in modo improprio. Per ovviare a questo incombente, alcuni ricercatori da diversi paesi si stanno adoperando per risolvere uno dei problemi più significativi legati alla diffusione di contenuti digitali: chi è l'autore di tale contenuto? Per rispondere a questa domanda ci avvaliamo del Machine Learning, più precisamente del Deep Learning[1]. Infatti, in ogni foto, o fotogramma di un video, è presente una componente di rumore dovuta a una particolare condizione dei circuiti elettronici della fotocamera, questo peculiare rumore è caratteristico del dispositivo e, dunque, può essere analizzato per associare quella specifica immagine al dispositivo che l'ha scattata.

Lo studio di queste informazioni è molto utile nell'ambito dell'analisi forense per istituire un nuovo metodo di contrasto verso i crimini informatici, anch'essi in notevole rialzo per il movimento che sta portando la maggior parte della popolazione alla digitalizzazione. Un esempio potrebbe essere la contribuzione a reperire prove attendibili in casi di abusi o cyberbullismo, oppure anche l'identificazione di altri atti illeciti come le infrazioni di copyright.

[1]Il Deep Learning è un campo del Machine Learning che si basa su diverse gerarchie di rappresentazione, per maggiori dettagli consiglio https://wikipedia.org/wiki/Deep_learning

Nello svolgimento del seguente elaborato analizzo e confronto le diverse soluzioni proposte da altri ricercatori, successivamente propongo delle metodologie sostitutive per valutare se è possibile migliorare le prestazioni e l'affidabilità del processo. Più precisamente suggerisco una modifica nel processo di selezione delle aree delle immagini da utilizzare per estrarre il rumore. Per questo scopo uso come riferimento due ricerche che vedono come autore principale la medesima persona: Guru Swaroop Bennabhaktula, difatti si basano su pressoché gli stessi concetti ma sono sviluppate con due tecniche alternative. I documenti a cui mi riferisco sono:

1. «Source Camera Device Identification from Videos» [3]

2. «Camera model identification based on forensic traces extracted from homogeneous patches» [2]

1.2 Metodologie alternative

L'utilizzo del rumore presente in ciascuna immagine, e di conseguenza dei soli valori dei pixel che la compongono, è solo uno degli svariati metodi che permettono di identificare la fotocamera sorgente, è altresì vero che è l'unico che permette di creare evidenze forensi. Questa sicurezza ci viene garantita dal fatto che il valore non può essere alterato, causa la sua natura: deriva dal processo di creazione dell'immagine ed è intrinseco al colore presente in ciascun pixel.

In caso venissero fatte delle alterazioni all'immagine, esistono dei metodi per identificare queste manomissioni, ne cito alcuni:

- Tra i più importanti, il *filtro mediano* utilizzato appunto per eliminare il rumore da un'immagine, viene risolto da Gao et al.[6].

- Simile al filtro mediano, esiste un'altra tecnica di eliminazione del rumore detta *filtro gaussiano*, identificabile grazie a Hwang e Rhee[10].

- Altra alterazione molto diffusa, soprattutto per la condivisione di contenuti, è la compressione JPEG analizzata nel dettaglio da Chennupati[5].

Come ho accennato prima, esistono altri metodi di riconoscimento del dispositivo sorgente, un esempio degno di nota sono quelli che si basano sui meta dati dell'immagine. Quest'ultimi si trovano sotto forma di intestazione EXIF[2] (Exchangeable Image File) nel file. Tuttavia, questi header possono essere agilmente alterati con l'utilizzo di software specifici, oppure potrebbero andare persi durante operazioni di modifica al file, una tra tanti è la compressione o la conversione ad un diverso formato. Per questo problema, i meta dati vengono considerati inaffidabili e non è possibile usarli in ambito forense, complice anche il fatto che non è in alcun modo possibile identificare se hanno subito una manomissione di qualunque tipo.

1.3 Motivazioni

Il già veloce aumento dell'utilizzo di dispositivi connessi a internet ha subito un'ulteriore impennata grazie al Covid19. Per l'appunto, quest'ultimo, ha contribuito a una massiccia digitalizzazione da parte di numerose persone, soprattutto in età non inclini alle tecnologie moderne: moltissime persone minorenni o anche anziani che non hanno mai utilizzato uno smartphone ad esempio. Questo fenomeno ha portato a un repentino aumento dei crimini informatici, sia in Italia che nel resto del mondo.

I dati a riguardo ci vengono riportati dalla Polizia Postale nei loro cospicui comunicati. Per quanto riguarda persone adulte e anziane, spicca sicuramente un aumento del 436% delle truffe sul web e fake news durante il 2020[14]. Non mancano però anche le minacce per la parte di popolazione più giovane, approcciatasi al mondo del web per poter usufruire della didattica a distanza, anch'essa vittima di innumerevoli crimini come cyberbullismo e altri abusi verso i minori. Tra i più significativi, la Polizia Postale riporta 2.083 casi di pedopornografia online con 202.927 Gigabytes di materiale sequestrato nel 2020[12], i quali si sono addirittura accentuati nel 2021 per un totale di 5.316 casi[13].

A seguito di queste informazioni disarmanti, risulta evidente che è necessario adoperarsi per cercare di contrastare il trend di crescita di suddetti reati: perché quindi non combattere tali abusi con la stessa tecnologia che li alimenta? È qui

[2]EXIF è uno standard per il formato di file immagine utilizzato dalle fotocamere digitali, vedi https://wikipedia.org/wiki/Exif

che possiamo sfruttare l'intelligenza artificiale per identificare chi ha prodotto e successivamente pubblicato le foto e i video imputati.

1.4 Obiettivi

Nello sviluppo di questo studio mi pongo l'obiettivo di proporre altri metodi di estrazione e di classificazione delle patch, alternativi a quelli presentati nei due documenti sopra citati. Per patch si intende una parte di immagine, nel mio caso un quadrato di alcuni pixel, che dovrà essere analizzato singolarmente per escludere la componente della scena contenuta nell'immagine integrale. Facendo ciò è possibile scegliere solo componenti che si presume contengano la migliore informazione di rumore possibile. I metodi proposti dagli altri autori si basano semplicemente sul prendere patch con un colore omogeneo, questa scelta rende però i segmenti presi inaffidabili: non è detto che contengano del rumore e quindi è necessario prenderne una grande quantità. Invece, se per selezionare i dettagli significativi dell'immagine usiamo un criterio di valutazione più consistente, possiamo naturalmente estrarne di meno e, di conseguenza, usare come input una collezione di dati con un numero maggiore di immagini.

Capitolo 2

Lavori correlati

Nel capitolo precedente, ho riportato quelli che considero, a mio avviso, un paio dei lavori più promettenti nell'ambito che sto trattando. Per aiutare il lettore nella comprensione dei capitoli successivi, riporto di seguito le informazioni rilevanti di ciascuno dei due documenti. Invito comunque, chi fosse interessato, alla lettura delle pubblicazioni ufficiali, trovate tutti i dettagli nella bibliografia.

2.1 Camera model identification based on forensic traces extracted from homogeneous patches

Innanzitutto presento il paper più recente, pubblicato il 10 Giugno 2022 da Guru Swaroop Bennabhaktula, Enrique Alegre, Dimka Karastoyanova e George Azzopardi[2]. Il loro progetto si pone l'obiettivo di trovare un metodo, come nel mio caso, per identificare uno specifico dispositivo partendo dalle immagini o dai frame dei video che avrebbe generato tale camera. Per raggiungere questo obiettivo, utilizzano una rete neurale convoluzionale completamente connessa. L'addestramento e i test vengono eseguiti con le sezioni di immagini omogenee prese da Dresden Image Database[7]. Per tutti i dettagli in merito al dataset vedi sezione 3.1.

Dato che è di nostro particolare interesse migliorare la procedura di estrazione delle patch dalle immagini, approfondisco nella sezione successiva la metodologia usata dagli autori di questo documento.

2.1.1 Estrazione patch omogenee

Il processo di estrazione delle patch consiste nella selezione delle regioni omogenee dell'immagine, quindi le parti che non variano particolarmente sotto il profilo del colore e possono essere usate per il nostro scopo. Per prima cosa si procede con la suddivisione dell'immagine in blocchetti di dimensione 128x128 pixel e uno stride di 0.25 volte la dimensione del blocco.

Per valutare l'omogeneità di ciascun blocco si procede con il determinare tre deviazioni standard, una per ogni canale di colore, quindi si individuano due soglie per i valori ottenuti: minore di 0.02 per escludere le zone non omogenee e maggiore di 0.005 per eliminare le patch sature. Le patch sature sono quelle che contengono perdita di dati perché il valore di molti pixel è troncato alla massima intensità. Infine, se le deviazioni standard di ciascun canale risultano valide per i limiti delle soglie, la patch si può considerare omogenea. Per rendere più efficiente il codice, gli autori, hanno pensato a un metodo che prevede di calcolare la deviazione standard dalla sua *Immagine integrale*[1] della porzione di foto.

La funzione esatta è riportata nel seguente Listato 2.1.

```
1 # Calcolo immagine integrale
2 i1 = torch.cumsum(torch.cumsum(patch, dim=0), dim=1)
3 i2 = torch.cumsum(torch.cumsum(patch ** 2, dim=0), dim=1)
4 # Calcolo deviazione standard
5 sum1 = # Sommatoria per l'immagine integrale 1
6 sum2 = # Sommatoria per l'immagine integrale 2
7 n = patch_size.width * patch_size.height # Area della patch
8 std_devs = torch.sqrt((sum2 - (sum1 ** 2) / n) / n)
```

Listato 2.1: Funzione calcolo deviazione standard da immagine integrale, paper 1.

Dopo aver classificato tutte le parti dell'immagine in analisi, è necessario selezionare quelle da usare nelle fasi successive. Per questo proposito viene usata la seguente logica: supponiamo di voler estrarre un numero di patch pari a p, se le parti omogenee sono più di p allora si estrapolano in modo da essere distribuite tra le regioni omogenee dell'immagine, se invece sono in numero minore si vanno via via a prendere quelle saturate e poi quelle non omogenee in ordine di varianza crescente.

[1] L'*Immagine integrale* in posizione (x,y) è la somma del valore dei pixel sopra e a sinistra di (x,y).

2.2 Source Camera Device Identification from Videos

Il secondo, ed ultimo, documento che cito perché inerente a questo argomento è stato prodotto il 14 Giugno 2021 da Guru Swaroop Bennabhaktula, Derrick Timmerman, Enrique Alegre e George Azzopardi[3]. In esso viene avanzata un'idea simile a quanto descritto in precedenza, infatti lo scopo è sempre quello di riconoscere il dispositivo che ha scattato una foto o un video tramite una rete convoluzionale preaddestrata. Anche qui è adottata la tecnica del rumore presente nell'immagine, ma la logica di applicazione è considerevolmente diversa, vedi paragrafo successivo. La differenza sostanziale rispetto all'elaborato precedente è che non viene effettuato un lavoro sulle singole parti dell'immagine, bensì viene eseguita la classificazione sull'intero frame. Attuando questo metodo si incontrano notevoli problemi con il contenuto dell'immagine dato che, non venendo escluso, potrebbe essere confuso dalla rete neurale come rumore. Un'altra particolarità è la scelta della collezione di dati usata per i test, a cui sottopongono due opzioni: il dataset Vision (sezione 3.3) o il database Dresden (sezione 3.1).

2.2.1 Metodologia

L'input di tutta la procedura, cioè i video dei dataset, è processato in tre fasi:

1. Estrazione frame: in cui viene proposto di estrarre N frame spaziati equamente all'interno del video. Quindi per ogni frame viene ritagliata la parte centrale (un rettangolo 480x800 pixel) per avere uniformità tra le immagini dei diversi dispositivi.

2. Classificazione dei frame: per prima cosa viene applicata una ConstrainedNet con alcuni filtri per ridurre l'impatto della scena sul rumore. Successivamente ogni frame viene classificato a seconda del suo possibile dispositivo originario.

3. Predizione del video: tutte le classificazioni dei frame vengono aggregate usando la regola conosciuta come "majority vote"[2].

[2]Majority vote rule è un metodo di fusione che consiste nell'assegnare a un pattern la classe che ha ricevuto più voti dai classificatori.

Capitolo 3

Dataset usati

Il dataset, trattandosi dell'insieme di immagini che vengono usate come addestramento e test per la rete, è una delle parti fondamentali per valutare la vera efficacia dell'implementazione. Scegliere un corretto set di immagini è altresì necessario per calcolare l'esatta accuratezza della rete in fase di inferenza, infatti una errata suddivisione delle immagini può comportare una altissima accuratezza nei test che poi però non rispecchierà la realtà. Nel caso di questo studio, le principali caratteristiche da tenere conto per un valido dataset sono:

1. Numero di marche: quante case produttrici di fotocamere e videocamere con le quali si desidera lavorare. Questo numero è rilevante per avere diverse casistiche, deve però rispecchiare quelle che sono le prospettive di utilizzo del sistema dopo la messa in produzione.

2. Numero di modelli per marca: simile al caso precedente, fa riferimento a quanti modelli vengono scelti per ciascuna casa produttrice.

3. Numero di dispositivi per modello: per verificare che effettivamente l'implementazione distingua con precisione lo specifico dispositivo è necessario avere almeno due dispositivi per ciascun modello.

4. Numero di immagini per dispositivo: le immagini, o i frame estratti dai video, devono essere in un numero sufficientemente grande e abbastanza diversificate da permettere il corretto riconoscimento dei pattern di rumore.

In questa implementazione vengono adottate le tre collezioni di dati usati dagli autori dei paper citati come riferimento. Più precisamente, essendo che ad oggi non

è più possibile ottenere il database Dresden, ho utilizzato le immagini di QUFVD per calcolare l'accuratezza di entrambi i documenti e le modifiche effettuate ad essi. Di seguito elenco le principali caratteristiche dei diversi dataset.

3.1 Dresden Image Database

Usato dal paper di riferimento più recente, sezione 2.1, risulta non essere più disponibile per il download nel sito web comunicato nella pubblicazione ufficiale[7]. Per ovviare a questo problema, ho riadattato la struttura del set di dati QUFVD in modo che sia utilizzabile dal codice implementato dagli autori che hanno prodotto il documento appena nominato, per la descrizione dettagliata faccio riferimento alla sezione successiva 3.2.

Di seguito riporto una breve analisi di come è strutturato il dataset in modo da poter anche confrontare i risultati che hanno ottenuto gli altri autori rispetto a quel set di immagini e quelli che ho ottenuto io usando QUFVD. Nella Tabella 3.1 riporto la lista di tutte le fotocamere impiegate, infatti non vengono considerati i modelli che hanno meno di due dispositivi, questo perché non sono sufficienti ad avere diversità tra gli stessi modelli e quindi non forniscono un caso realistico.

3.2 QUFVD

Il dataset QUFVD (Qatar University Forensic Video Database [1]) viene adottato dagli autori del paper meno recente (sezione 2.2). In questo sviluppo, oltre ad essere usato per estrarre i risultati dall'appena citato documento, riadatto la struttura dei file in modo da poterlo usare come sostituto a Dresden Image Database nella verifica del corretto funzionamento dello modifiche proposte.

Per prima cosa riporto le medesime informazioni, Tabella 3.2, per il confronto con il dataset precedente. Si può subito notare che, benché il numero di dispositivi sia minore, le immagini presenti in questo dataset siano notevolmente maggiori. Nonostante QUFVD sia relativamente migliore rispetto al punto 4 dell'analisi fatta a inizio capitolo, Dresden risulta più consistente dal punto di vista dei punti da 1 a 3, questa differenza sicuramente ci fornirà dei valori di accuratezza diversi da quelli originali, però ci permetterà anche di confrontare quale dei due approcci è migliore: più diversificazione nei dispositivi o più immagini per la stessa fotocamera?

Tabella 3.1: Dettaglio dei modelli di fotocamera presenti in Dresden Image Database che hanno almeno due dispositivi.

Produttore	**Modello**	**N. Dispositivi**	**N. Immagini**
Canon	Ixus 70	2	542
Casio	EX-Z150	5	925
Fujifilm	FinePix J50	3	495
Kodak	M1063	5	2160
Nikon	Coolpix S710	5	916
Nikon	D70	4	715
Nikon	D200 Lens	2	712
Olympus	1050SW	5	1027
Panasonic	DMC-FZ50	3	561
Pentax	Optio A40	4	629
Praktica	DCZ 5.9	5	973
Ricoh Capilo	GX100	5	715
Rollei	RCP+7325XS	3	581
Samsung	L74wide	3	562
Samsung	NV15	3	566
Sony	DSC-H50	2	308
Sony	DSC-T77	4	565
Sony	DSC-W170	2	272
	TOTALE	**65**	**13224**

Tabella 3.2: Dettaglio dei modelli dei telefoni presenti nel dataset QUFVD.

Marchio	**Modello**	**N. Dispositivi**	**N. Video**	**N. Frame**
Samsung	Galaxy A50	2	600	7436
Samsung	Note9	2	600	7918
Huawei	Y7	2	600	7272
Huawei	Y9	2	600	8157
Apple	iPhone 8 Plus	2	600	8071
Apple	iPhone XS Max	2	600	7967
Nokia	5.4	2	600	6881
Nokia	7.1	2	600	7720
Xiaomi	Redmi Note8	2	600	7374
Xiaomi	Redmi Note9 Pro	2	600	7726
	TOTALE	**20**	**6000**	**76522**

3.2.1 Conversione da QUFVD a Dresden

Dopo aver analizzato le differenze sulla base del contenuto, considero quelle relative all'organizzazione dei file. Infatti, per utilizzare il software, è necessario che l'input sia salvato in uno specifico formato. Per conoscenza del lettore riporto le modifiche che ho apportato alla struttura delle cartelle per poter utilizzare il dataset QUFVD come sostituto al database Dresden. La presente sezione è utile anche se si vuole adottare qualsiasi altro dataset in quanto l'organizzazione dei file e dei loro nomi deve essere esattamente:
marca_modello_n.dispositivo/marca_modello_n.dispositivo_n.immagine.jpg .

Di seguito riporto un esempio di come si deve presentare l'input per essere correttamente analizzato:

```
/
├── Marca1_Modello1_1/
│   ├── Marca1_Modello1_1_100.jpg
│   ├── Marca1_Modello1_1_200.jpg
│   └── ...
├── Marca1_Modello1_2/
│   └── ...
└── ...
```

Dato che la struttura del dataset QUFVD è *marca-modello/Device[1|2]/marca-modello-n.dispositivo(n.video)-n.immagine.jpg*, trascrivo qui sotto lo script che ho prodotto per la conversione dei nomi dei file.

```
1 f = # nome file
2 f = f. replace("-", "_")
3 # Estraggo il numero del video inserito tra le parentesi
4 video_id = f.split("(")[1].split(")")[0]
5 # Estraggo il numero del frame inserito dopo le parentesi
6 frame_id = f.split(")")[1].split("_")[1]
7 # Unisco nel formato
     marca_modello_n.dispositivo_video-frame.jpg
8 nome = f.split("(")[0] + video_id + "-" + frame_id
```

Listato 3.1: Funzione conversione nome file da QUFVD a Dresden dataset.

3.3 Vision

L'ultima collezione di video che viene riportata dagli autori citati nei capitoli precedenti è chiamata Vision. La differenza principale che lo caratterizza è che questo dataset contiene più di trentamila immagini e quasi duemila video, ciascuno di essi presente in diversi formati[16]:

- Nativo: estratto direttamente dal risultato prodotto dal dispositivo;
- Social Network: condivisi tramite social quali Facebook, YouTube e WhatsApp.

Oltre ai diversi formati, le immagini sono anche suddivise in due macro tipologie:

- Flat (piatto): immagini o video che contengono scene fisse, per esempio il cielo o un muro con colore uniforme;
- Indoor/Outdoor: sono evidenziati sia media presi all'aperto o in luogo chiuso.

La sua costruzione peculiare lo rende particolarmente adatto per l'obiettivo di questo elaborato, infatti al giorno d'oggi la maggior parte dei video vengono caricati sui Social Network e quindi è interessante capire come determinati processi influiscano sul riconoscimento del dispositivo sorgente.

Terminati gli aspetti positivi, è doveroso menzionare anche quelli negativi. Un grave problema di questa collezione è che non rispecchia la realtà, infatti non soddisfa il punto 3 dei requisiti che ho precedentemente individuato: ci sono solo un paio di modelli che hanno più dispositivi, di tutti gli altri è presente un solo esempio. Di conseguenza, non essendo simile a un possibile caso in inferenza, faccio riferimento ai risultati ottenuti dagli altri autori con Vision ma non lo considero per ulteriori implementazioni. Sarebbe rimarchevole integrare il dataset con altri media di differenti dispositivi ed estratti con la stessa logica, lo renderebbe sicuramente la collezione di video più utilizzata per questo genere di studi.

Capitolo 4

Rumore dell'immagine

Prima di poter procedere con l'implementazione, è necessario presentare un'analisi approfondita del fenomeno che viene denominato rumore dell'immagine. Questo evento è definito come una variazione, rispetto a quello che è il reale valore, delle due componenti principali di una foto: luminosità e colore presente in immagini o frame di video. Generalmente è una conseguenza dello stesso rumore generato dai componenti elettronici del dispositivo, per esempio i circuiti o il sensore della fotocamera.

4.1 Tipi di rumore

Il rumore di un'immagine può presentarsi sotto diverse forme, elenchiamo qui sotto quelle più rilevanti per lo scopo di questo esposto:

- Rumore gaussiano: principalmente generato nel momento di acquisizione dell'immagine, dipende dai livelli di illuminazione, dalla temperatura del sensore e dai circuiti dello stesso. Si presenta soprattutto nelle zone scure e sui canali dei colori blu e verde dato che sono quelli dove è più necessaria l'amplificazione del segnale. [19]

- Rumore sale e pepe: è detto anche impulsivo perché si evidenzia con scarsi pixel scuri in zone chiare dell'immagine, o viceversa. Può essere causato da errori di trasmissione o dal convertitore analogico-digitale. [20]

- Rumore scatto: si tratta del rumore più presente nelle zone dell'immagine particolarmente luminose. Infatti è causato dalle fluttuazioni quantistiche statiche, ovvero le variazioni nel numero di fotoni rilevati a una certa esposizione. Questo è un esempio di rumore casuale che quindi non può essere usato per l'identificazione della camera sorgente. [21]

4.2 Cause del rumore

Dopo aver definito con precisione le tipologie del rumore, possiamo identificare anche le principali cause scatenanti: [15]

- Scarsa luminosità: il sensore del dispositivo, in condizioni di luce scarsa e di conseguenza di ISO maggiore, sarà indotto a colmare con del rumore le parti dove non è stato possibile acquisire le informazioni, viceversa, quando l'ambiente è sufficientemente illuminato, la luce andrà quasi completamente a coprire quello che è il rumore elettronico.
- Alta sensibilità: si tratta di un procedimento che amplifica i risultati prodotti dal sensore, essendo un normale processo di amplificazione, oltre alle informazioni della luce, incrementa anche quello che è il rumore.
- Sensore di dimensioni ridotte: unito al fatto di richiedere una sempre più alta risoluzione, causa l'accumulo di pochi fotoni sul sensore e di conseguenza l'introduzione di più rumore.
- Inferenza: oltre ai componenti della fotocamera, il rumore può essere anche derivante da fenomeni esterni, per esempio radiazioni cosmiche o interferenze con altri segnali.

4.3 Rumore a pattern fisso

Il rumore, in alcuni casi, potrebbe essere causato da eventi non costanti nel tempo e che quindi, rendendolo casuale, risulta non funzionale per il nostro scopo. Per analizzare la relazione tra il dispositivo e il suo rumore è necessario valutare il Fixed-Pattern Noise. "FPN è un termine generale che identifica una non uniformità laterale temporalmente costante (formando un pattern costante) in un sistema di imaging con più elementi di rivelatore o immagine (pixel)"[9]. Quest'ultima frase

sta a significare che il rumore a pattern fisso è un tipo di rumore caratterizzato da una variazione dei pixel chiari e scuri, ed è dovuto esclusivamente dai componenti elettronici interni del dispositivo, di conseguenza rimane invariato nel corso del tempo. Non si esclude però il fatto che si possano comunque formare delle minime variazioni nel corso di vita della fotocamera che portano a un, seppur minimo, cambiamento del FPN, per esempio potrebbe essere causato dall'usura del sensore col tempo o anche dai movimenti dei componenti conseguentemente a un urto.

Le due componenti principali del rumore a pattern fisso sono PRNU e DSNU, le analizzo nel dettaglio nelle due sottosezioni specifiche. [11]

4.3.1 Photo Response Non-Uniformity (PRNU)

Il PRNU è considerato da molti la "impronta digitale" delle fotocamere dato che è costante nel tempo e specifico per ciascun dispositivo, ma vediamo nel dettaglio da cosa è verificata questa attribuzione.

Quando si parla di PRNU si fa riferimento all'alterazione che avviene durante il processo di conversione del segnale da analogico a digitale. Il segnale elettronico, estratto dalla conversione dei fotoni, viene analizzato lato software per il calcolo del relativo valore digitale, a seconda del fattore di conversione si potrebbe avere che il dispositivo in alcuni punti non riesce a tradurre le minime oscillazioni dei livelli dei fotoni, motivo per cui viene applicata una approssimazione nei punti meno sensibili creando delle differenze rispetto ai pixel vicini. Per questo tipo di pattern, essendo dipendente dal livello di illuminazione, è preferibile confrontare immagini con la stessa quantità di luce. Il PRNU è condizionato principalmente da due effetti: [17]

1. Brillantezza: essendo il PRNU dipendente dalla luminosità, il risultato può variare a seconda della quantità di luce che colpisce il sensore della fotocamera. Infatti il rumore è praticamente inesistente nelle zone ad altissima luminosità, è scarso nelle zone prive di illuminazione e si nota maggiormente nelle zone scure dell'immagine.

2. Frequenza: le immagini aventi dettagli ad alta frequenza creano delle distorsioni nell'estrazione del PRNU, questo perché il contenuto potrebbe essere appreso come rumore dalla rete causando degli errori di classificazione nelle fasi successive. Vedi un esempio in Figura 4.1.

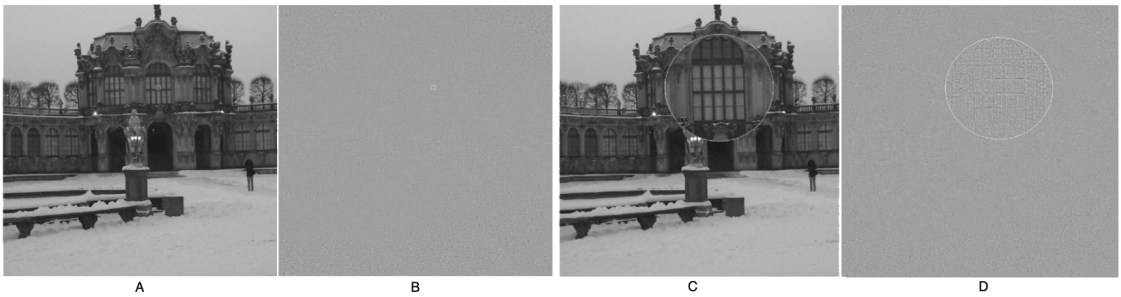

Figura 4.1: Esempio di come influisce la frequenza dell'immagine sul PRNU, estratto da [17] e rielaborato. Si può subito notare come l'immagine D, nella parte di foto con uno zoom applicato, presenti una miglior stima del PRNU.

4.3.2 Dark Signal Non-Uniformity (DSNU)

Il rumore in una fotocamera ha un andamento fluttuante e può capitare che le variazioni casuali negative comportino un valore del pixel sotto lo zero, causando così innumerevoli errori nei processi successivi di salvataggio e visualizzazione. Per far fronte a questo problema, i sistemi moderni aggiungono un *bias*[1] a ogni pixel, non è altro che un valore arbitrario che permette di assegnare un valore al pixel anche se il segnale dal sensore è nullo (non c'è luce). Il DSNU si presenta a seguito di questo processo, proprio perché per un fattore fisico di come è costruita la camera, è impossibile che il bias abbia lo stesso esatto valore in ogni pixel e quindi si va a creare questo pattern di differenze facilmente visibile in assenza di luce. Riporto un esempio in Figura 4.2.

A differenza del PRNU, il DSNU è indipendente dalla luminosità, ma è invece interessato dalla temperatura e dal tempo di esposizione, questo perché i pixel caldi hanno una sensibilità diversa di quelli freddi, d'altra parte un maggior tempo di esposizione comporta un aumento della temperatura.

[1] In questo caso uso il termine *bias* per indicare un valore costante che viene sempre aggiunto a prescindere dall'esito delle altre operazioni ed è necessario ad aggiustare il risultato.

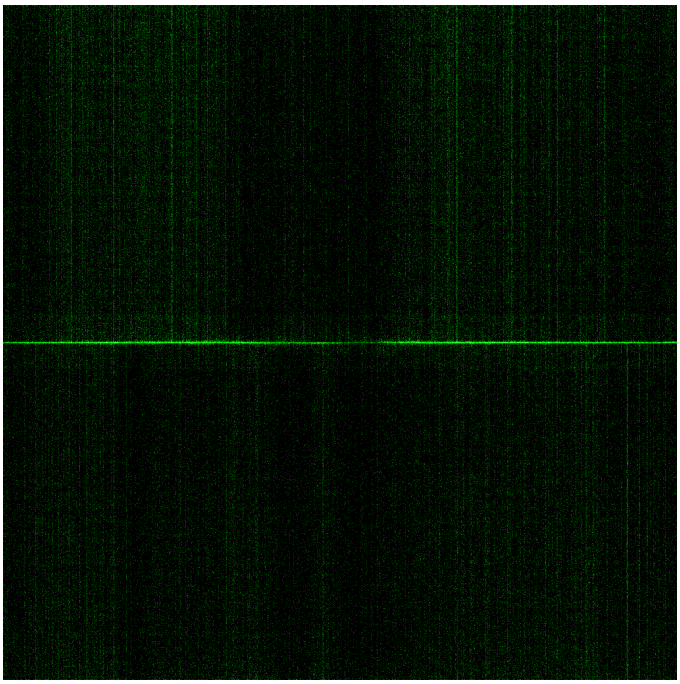

Figura 4.2: Esempio di rumore DSNU con una media di 100 immagini catturate con nessuna illuminazione e 0 ms di esposizione.

4.4 Considerazioni

Dalla precedente analisi delle diverse tipologie di rumore, ho appreso che il rumore, può essere sì influenzato da fenomeni esterni, come la temperatura e la presenza o meno di luce, ma deriva anche da come le componenti elettroniche della fotocamera reagiscono a tali fenomeni. Attraverso il Deep Learning e l'uso di patch significative come input, la rete neurale è in grado di escludere quello che è il rumore casuale, ovvero quello non provocato dal dispositivo ma dovuto a una particolare condizione dei fotoni in quel preciso istante. Per ottenere la maggior accuratezza possibile dalla rete neurale, è necessario usare come input un grande quantitativo di immagini così da poter estrarre da esse l'informazione del rumore che persiste tra i diversi frame. Dato che il processo di estrazione delle patch analizza le immagini ed estrae p^2 parti delle immagini di dimensione 128x128 pixel, il risultato di tale procedura crea un input per la rete con numerose componenti da analizzare e quindi comporta un

[2]I dettagli della procedura sono stati trattati nella sottosezione 2.1.1

notevole aumento dei tempi di computazione e di spazio.

Per ridurre il problema citato in precedenza, e quindi poter usare più frame come dataset per l'addestramento, dopo aver studiato il rumore ho riscontrato che la graduatoria delle patch già estratte dagli autori citati si può integrare con i seguenti accorgimenti:

1. Luminosità: non considerare le zone con altissima luminosità perché non contengono rumore, invece considerare le zone più scure e che si aggirano intorno al 25% di saturazione.

2. Frequenza: prediligere le parti di immagine che hanno una più bassa frequenza di contenuto in modo che non possa interferire con quella che è l'informazione del rumore.

3. Colore: dato che il colore blu è per costruzione è soggetto ad una maggiore amplificazione, dare più importanza ai risultati ottenuti con gli algoritmi applicati al canale di quel colore. In ordine poi viene considerato il canale del colore verde e infine il rosso.

Capitolo 5

Metodologia

Terminato lo studio approfondito del rumore e di come si comporta in diverse condizioni, espongo come ho implementato praticamente quanto emerso nel precedente Capitolo 4. Nella sezione 5.1 riporto il codice sviluppato per l'estrazione delle patch, invece nella sezione 5.2 elenco nel dettaglio tutti i parametri scelti per il perfezionamento dell'algoritmo.

5.1 Estrazione patch

In questo elaborato, la variazione principale che propongo riguarda la selezione delle patch omogenee da usare come dati per l'addestramento della rete e come input per il riconoscimento del dispositivo. La modifica si basa sul codice proposto da Bennabhaktula et al. nel suo GitHub ufficiale[1], ad ogni modo tutte le informazioni riguardanti le sue logiche sono riportate nella sezione 2.1. Per ridurre l'impatto sul resto del software, ho pensato a tre funzioni che posso essere aggiunte prima del salvataggio delle patch e che servono a produrre un sistema di ranking e quindi di selezione alternativa a quello già proposta. Si dividono in: due funzioni per il calcolo dei valori specifici per la singola componente, quali esposizione e frequenza del contenuto, e una funzione di unione e classificazione dei risultati.

[1] https://github.com/bgswaroop/scd-images

5.1.1 Calcolo Fast Fourier Transform

Inizialmente approfondisco quella che è la funzione per ottenere la frequenza del contenuto dell'immagine, necessità emersa dal punto 2 delle considerazioni per lo studio del rumore. Per questo calcolo mi sono affidato alla Trasformata di Fourier che permette di "convertire il segnale dal suo dominio originale alla rappresentazione nel dominio della frequenza"[18]. Dato che le immagini, e quindi le patch, vengono gestite nel codice tramite le librerie Pillow e Numpy, ho usato la funzione *numpy.fft.fft2*[2] già presente in quest'ultime. Trascrivo di seguito nel Listato 5.1 come ho previsto che venga sfruttata la FFT per l'operazione.

```
1  def get_fft_patch(patch):
2    # Calcolo la Fast Fourier transform
3    rows, cols = patch.shape
4    f = np.fft.fft2(patch)
5    fshift = np.fft.fftshift(f)
6    crow, ccol = int(rows/2), int(cols/2)
7    # Lavoro con sezioni di FFT_DIM pixel
8    fshift[crow-int(FFT_DIM/2):crow+int(FFT_DIM/2),
        ccol-int(FFT_DIM/2):ccol+int(FFT_DIM/2)] = 0
9    f_ishift = np.fft.ifftshift(fshift)
10   img_back = np.fft.ifft2(f_ishift)
11   img_back = np.abs(img_back)
12   v = np.average(img_back)
13   # Considero solo valori entro il limite
14   return v if v <= FFT_THRESHOLD else 0
```

Listato 5.1: Funzione che calcola la Fast Fourier transform di una patch.

Volendo rendere più chiaro al lettore qual'è il risultato del calcolo della trasformata di Fourier, allego Figura 5.1 da cui si può evincere che le aree più chiare siano quelle con una maggiore variazione in quello che è il contenuto della foto.

5.1.2 Calcolo valore esposizione

Il punto 1 della sezione 4.4, che fa riferimento al fatto che il rumore è più presente nelle aree scure dell'immagine, ma non nere, l'ho trattato creando una funzione

[2] https://numpy.org/doc/stable/reference/generated/numpy.fft.fft2.html

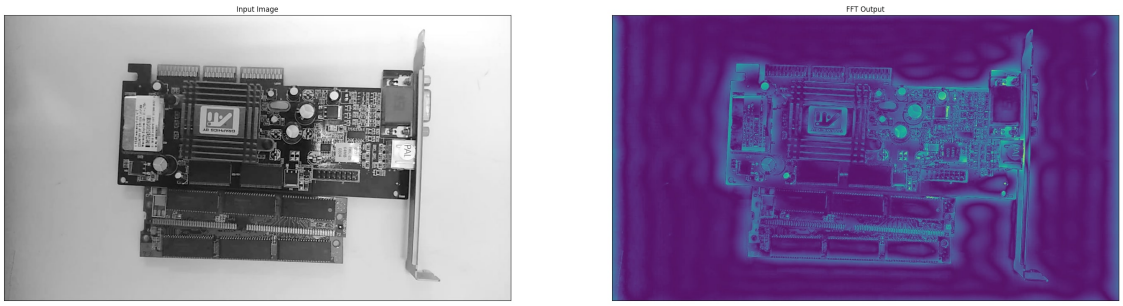

Figura 5.1: Esempio risultato applicazione trasformata di Fourier su un'immagine campione.

che calcola quanto si discosta l'esposizione della foto dal valore desiderato. Dato che la patch è già convertita in scala di grigi, si può direttamente avvalersi della differenza tra il valore della media dei pixel e il 25% di 255 (colore bianco e valore massimo che può assumere un pixel). Infine per normalizzare il risultato è necessario dividere il valore per una costante pari a 0.75, trattandosi del numero massimo che può restituire la funzione precedente. Riporto i particolari in Listato 5.2.

```
1 def get_exposure_patch(patch):
2   exp = 1 - abs(np.average(patch/255) -
       EXPOSURE_TARGET)/(1-EXPOSURE_TARGET)
3   return exp
```

Listato 5.2: Funzione che calcola la differenza di esposizione rispetto al valore desiderato.

5.1.3 Ordinamento e selezione patch

In conclusione, ho stabilito una funzione che sfrutti i risultati delle due precedenti fasi e che li fonda tenendo conto dei differenti pesi.

Come primissima cosa, commuto la patch in scala di grigi così da semplificare le fasi seguenti. Successivamente, converto la deviazione standard generata sui tre canali dei colori in modo che sia anch'essa un valore unico, in questa fase applico il punto 3 riguardante le considerazioni sul rumore per dare più importanza al colore blu. Dunque calcolo i due valori di esposizione e frequenza della specifica patch e li aggiungo con i loro pesi, il tutto normalizzando i risultati per ottenere un valore compreso tra 0 e 1. Concludo ordinando tutte le patch in ordine di punteggio

decrescente e quindi la funzione restituisce le prime N richieste. Inserisco il codice in Listato 5.3.

```
def patches_ranking(num_patches, patches, scores):
  res = []
  fft_scores = []
  for i in range(len(patches)):
    # Converto in scala di grigi
    grayValue = 0.07 * patches[i][:, :, 2] + 0.72 * \
        patches[i][:, :, 1] + 0.21 * patches[i][:, :, 0]
    patch = grayValue.astype(np.uint8)
    # STD RGB
    score = (1 - (scores[i][0]*RED_WEIGHT + scores[i][1] *
             GREEN_WEIGHT +
                scores[i][2]*BLUE_WEIGHT))*WEIGHT_STD
    fft_scores.append(get_fft_patch(patch))
    # Punteggio esposizione
    score += get_exposure_patch(patch)*WEIGHT_EXPOSURE
    res.append([scores[i], patches[i], score])
  fft_max = max(fft_scores)  # Normalizzazione
  if fft_max > 0:
    for i in range(len(patches)):
      # Punteggio FFT
      res[i][0] += (1 - fft_scores[i]/fft_max)*WEIGHT_FFT
  # Ordino per il punteggio dato
  res.sort(key=lambda val: val[2], reverse=True)
  return [(x[0], x[1]) for x in res[:num_patches]]
```

Listato 5.3: Funzione che crea i punteggi per ciascuna patch e seleziona quelle più adatte secondo le logiche definite.

5.2 Parametri utilizzati

Negli estratti di codice sorgente proposti nella sezione precedente sono presenti dei parametri, cioè le costanti riconoscibili poiché sono identificate da tutte lettere maiuscole. Le appena citate costanti possono essere parametrizzate con diversi criteri e permettono di poter aggiustare l'esecuzione della rete neurale per raggiungere la migliore accuratezza possibile. Io propongo di impiegare i seguenti valori:

- Numero di patch omogenee estratte con la logica descritta dal documento di partenza e che poi vengono classificate con il nuovo algoritmo: 200
- EXPOSURE_TARGET: valore esposizione a cui ritengo che il rumore sia più considerevole, pari a 63.75 (25% della saturazione).
- Parametri relativi al calcolo della trasformata di Fourier:
  - FFT_DIM: dimensione della sezione analizzata, può essere regolata per rendere più preciso il calcolo della frequenza, io considero adatto usare 20x20 pixel.
  - FFT_THRESHOLD: soglia di valore oltre il quale la frequenza non è più considerata accettabile perché troppo elevata. Questo numero è strettamente dipendente dalla dimensione scelta prima, a seguito di alcuni test e verifiche io consiglio di non impiegare frequenze superiori a 5.
- Pesi attribuiti alle diverse classificazioni:
  - WEIGHT_FFT - Fast Fourier transform: 0.6
  - WEIGHT_STD - deviazione standard: 0.2
  - WEIGHT_EXPOSURE - esposizione o lucentezza: 0.2
  - Deviazione standard sui canali dei colori: BLUE_WEIGHT 0.5, GREEN_WEIGHT 0.3, RED_WEIGHT 0.2

Infine, elenco come riferimento anche i parametri di cui mi sono avvalso per l'addestramento della rete convoluzionale:

- Numero massimo di epoche: 45
- Learning rate: 0.01
- Batch size: 30
- Numero di patch: 50
- Fold: 0, 1, 2, 3, 4

5.3 Altri miglioramenti

Durante lo studio del codice e l'implementazione delle modifiche sono emerse anche alcune idee per un possibile miglioramento delle prestazioni nella prima scrematura delle parti omogenee dell'immagine. Quella parte del processo richiede molto tempo al momento del primo addestramento, infatti è necessario analizzare ogni immagine di tutte le fotocamere, ma in inferenza questa operazione verrà eseguita solo quando si vuole aggiungere un nuovo dispositivo. Per questo motivo non è di fondamentale importanza avere prestazioni elevate, ci tengo comunque a riportare i concetti qui sotto per una eventuale evoluzione futura.

La prima tra tutte si basa sul fatto che, al momento della scelta se una parte dell'immagine è omogenea o no, viene calcolata la deviazione standard per ciascun canale di colore, si tratta però di un'operazione molto lunga e dispendiosa. Una soluzione veloce sarebbe quella di mutare l'immagine in scala di grigi e calcolare una sola deviazione standard. Facendo alcuni test preliminari ottengo che il risultato è estremamente simile a quello dato dall'immagine a colori, bensì il tempo di elaborazione è più che dimezzato. Un'altra proposta potrebbe essere quella di decidere se un'area dell'immagine è omogenea o no tramite il calcolo della sua entropia e quindi la generazione della tecnica che è detta *Immagine derivativa*[4].

Capitolo 6

Conclusioni

6.1 Risultati addestramento

Il primo risultato che riporto come riferimento l'ho ricavato eseguendo l'addestramento della rete neurale con i parametri indicati nella sezione 5.2. L'esecuzione è stata svolta utilizzando il codice proposto da Bennabhaktula et al.[2] e le patch estratte da Dresden Image Database (sezione 3.1) seguendo la metodologia che ho proposto nei capitoli precedenti. I valori significativi ottenuti sono mostrati in Tabella 6.1 con i particolari dei diversi *fold*. Infatti l'algoritmo, per ottenere una migliore generalizzazione, usa la tecnica del *5-fold cross-validation*[1] sulla collezione di dati in ingresso e, quindi, tramite "Fold N" ho specificato quale è stato utilizzato per ottenere quel risultato.

Tabella 6.1: Risultati addestramento rete neurale con metodo proposto in questo documento.

	Fold 0	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**
Tempo Esecuzione (h)	15	13.26	12	11.25	11
Accuratezza Test	0.966	0.97	0.965	0.97	0.965
Accuratezza Validation	0.99	0.993	0.99	0.993	0.99
F1	0.99	0.993	0.99	0.993	0.99
Recall	0.99	0.993	0.99	0.993	0.99

[1] *K-Fold Cross-Validation* è una tecnica che permette di dividere il dataset in K porzioni che poi verranno alternate tra addestramento e test.

Tramite la Figura 6.1 qui inserita, do una visione dettagliata di come varia l'accuratezza durante la fase di validazione dell'addestramento. Nel grafico mostro gli andamenti dei diversi *fold* e nello specchietto riporto il dettaglio dei valori finali di ciascuno di essi.

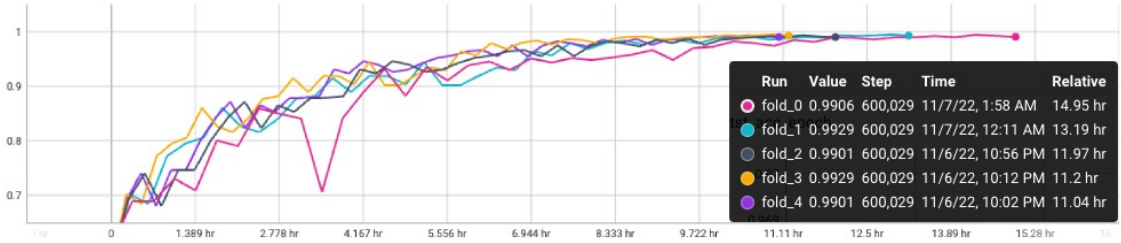

Figura 6.1: Grafico che mostra l'andamento dell'accuratezza in fase di validazione dell'addestramento con il metodo proposto in questo documento.

6.2 Risultati selezione patch

L'obiettivo di questo elaborato è quello di proporre un metodo di classificazione per le patch omogenee estratte tramite l'algoritmo realizzato da Bennabhaktula et al. Applicare una classificazione più consistente significa, di conseguenza, poter ridurre la dimensione delle parti di immagine selezionate. Sfruttando queste logiche si possono ridurre i tempi computazionali per ciascun dispositivo, ma soprattutto si possono usare più foto e più video per eseguire l'addestramento della rete neurale.

Allego la Figura 6.2 dove mostro un esempio pratico del risultato ottenuto applicando l'algoritmo proposto nel capitolo precedente. La scelta delle aree da considerare come input per il processo di classificazione è dimostrata dal fatto che ho evidenziato ciascuna patch riquadrandola con un contorno colorato e diverso per ciascuna così da poterle distinguere anche se sovrapposte. Per la presentazione è mostrata in alto (immagine A) la foto originale usata come esempio per il confronto, ho scelto un'immagine con un soggetto principale molto elaborato e uno sfondo non uniforme per evidenziare meglio le scelte fatte. In basso a sinistra (immagine B) ho inserito l'output dell'algoritmo di partenza. Infine, a destra (immagine C) propongo il risultato di quanto sottoposto da me.

La differenza che salta subito all'occhio è che nell'immagine C sono state selezionate molte meno parti: comportamento perfettamente in linea con l'obiettivo da

raggiungere. La correttezza è sostenuta dal fatto che molte delle frazioni dell'immagine selezionate dall'algoritmo a cui ho applicato i cambiamenti si trovano in aree con altissima luminosità e quindi, per quanto riportato al punto 1, non possono contenere rumore poiché il sensore è saturato dai fotoni in ingresso. Altra accortezza è che viene dato poco peso alle componenti ad alta intensità per dare spazio a quelle più uniformi e, in ordine, quelle con saturazione circostante al 25% della capacità della fotocamera.

6.3 Lavori futuri

Per le modifiche ho considerato solo l'implementazione "flat" descritta nel documento più recente, quindi la classificazione basata sui dispositivi e non quella gerarchica che permette di identificare le patch partendo dai marchi delle fotocamere e arrivando infine al dispositivo. Dato che il documento citato sviluppa entrambe le metodologie, per un lavoro futuro è possibile prendere in considerazione anche di applicare le idee di questo elaborato al classificatore gerarchico.

Durante lo studio delle possibili soluzioni per risolvere il problema del riconoscimento del dispositivo sorgente sono emerse delle alternative che però si basano su una diversa tecnica di estrazione del rumore da un'immagine. A differenza di quanto proposto da me e dagli autori dei documenti citati, si potrebbe pensare ad implementare una logica che punta ad estrarre il rumore di un'immagine attraverso tecniche di denoising[2]. Il principio alla base di queste idee è di applicare all'immagine, o nel nostro caso a parti di essa, un algoritmo di rimozione del rumore, successivamente sottrarre l'immagine appena estratta a quella originale e usare il risultato come input per la rete neurale.

Altre tecniche potrebbero vedere come protagonista il Machine Learning e diverse tecniche di selezione delle componenti omogenee all'interno dei frame. Quelle più semplici potrebbero prevedere l'impiego di una rete neurale precedentemente addestrata per il riconoscimento di oggetti. Altre, più sofisticate, come il "Flood fill"[3], giusto per citarne una tra tante. Per un ulteriore approfondimento consiglio l'articolo di Goltsev, Gritsenko e Húsek[8].

[2]Per approfondire le diverse tecniche di rimozione del rumore faccio riferimento a: https://link.springer.com/article/10.1007/s40747-021-00428-4

[3]https://en.wikipedia.org/wiki/Flood_fill

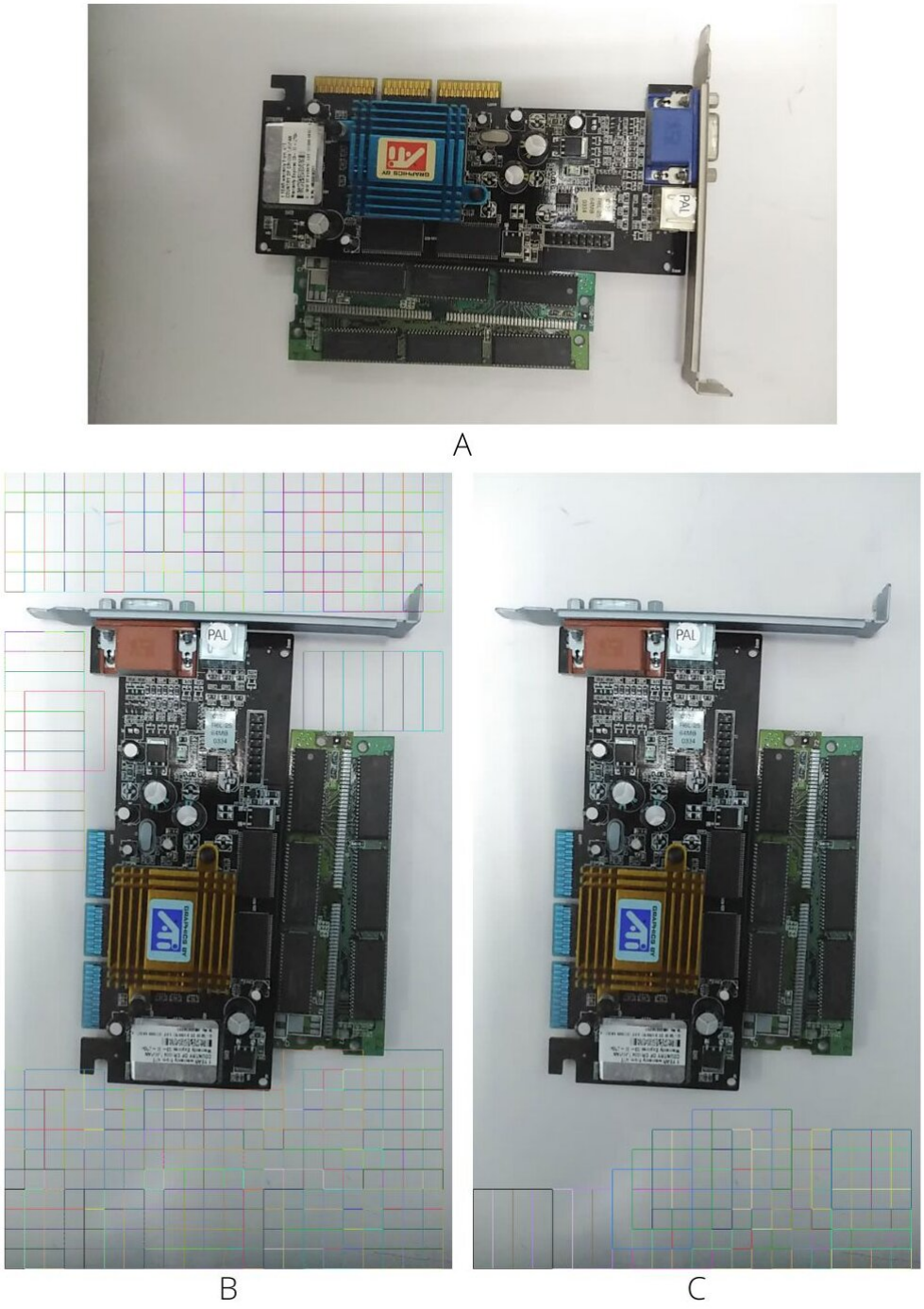

Figura 6.2: Differenza tra le patch selezionate dai diversi algoritmi: in alto l'immagine di prova in input, a sinistra l'output con la logica originale e a destra quello proposto in questo documento. Ogni quadrato colorato nell'immagine è una patch selezionata.

Bibliografia

[1] Younes Akbari et al. «A New Forensic Video Database for Source Smartphone Identification: Description and Analysis». In: *IEEE Access* 10 (2022), pp. 20080–20091. DOI: 10.1109/ACCESS.2022.3151406.

[2] Guru Swaroop Bennabhaktula et al. «Camera model identification based on forensic traces extracted from homogeneous patches». In: *Expert Systems with Applications* 206 (2022), p. 117769. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.117769. URL: https://www.sciencedirect.com/science/article/pii/S0957417422010430.

[3] Guru Swaroop Bennabhaktula et al. «Source Camera Device Identification from Videos». English. In: *SN Computer Science* 3.4 (giu. 2022). ISSN: 2661-8907. DOI: 10.1007/s42979-022-01202-0.

[4] Giuseppe Pio Cannata. *Image derivative*. https://towardsdatascience.com/image-derivative-8a07a4118550. [Online; accessed 03-November-2022]. 2021.

[5] Om Sai Teja Chennupati. «A structured approach to JPEG tampering detection using enhanced fusion algorithm». Tesi di laurea mag. Computer Science: Blekinge Institute of Technology, gen. 2021. URL: https://www.diva-portal.org/smash/get/diva2:1535158/FULLTEXT02.

[6] Hang Gao et al. «Robust detection of median filtering based on combined features of difference image». In: *Signal Processing: Image Communication* 72 (2019), pp. 126–133. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2018.12.014. URL: https://www.sciencedirect.com/science/article/pii/S0923596518308464.

[7] Thomas Gloe e Rainer Böhme. «The 'Dresden Image Database' for Benchmarking Digital Image Forensics». In: *Proceedings of the 2010 ACM Symposium on Applied Computing*. SAC '10. Sierre, Switzerland: Association for Computing Machinery, 2010, pp. 1584–1590. ISBN: 9781605586397. DOI: 10.1145/1774088.1774427. URL: https://doi.org/10.1145/1774088.1774427.

[8] Alexander V. Goltsev, Vladimir Gritsenko e Dusan Húsek. «Extraction of homogeneous fine-grained texture segments in visual images». In: *Neural Network World* 27 (2017), pp. 447–477.

[9] HB Railway. *Rumore a pattern fisso*. https://hbrailway.com/it/rumore-a-pattern-fisso/. [Online; accessed 21-October-2022]. 2022.

[10] Jae-Jeong Hwang e Kang Hyeon Rhee. «Gaussian filtering detection based on features of residuals in image forensics». In: *2016 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)* (2016), pp. 153–157. DOI: 10.1109/RIVF.2016.7800286.

[11] *Pattern Noise: DSNU and PRNU*. https://www.photometrics.com/learn/advanced-imaging/pattern-noise-dsnu-and-prnu. [Online; accessed 22-October-2022].

[12] Polizia postale. *Operazione della Polizia postale contro la pedopornografia on line*. https://www.interno.gov.it/it/notizie/operazione-polizia-postale-contro-pedopornografia-line. [Online; accessed 10-October-2022]. 2020.

[13] Polizia postale. *Pedopornografia e pedofilia in aumento nel 2021: il dossier della Polizia postale*. https://www.interno.gov.it/it/notizie/pedopornografia-e-pedofilia-aumento-nel-2021-dossier-polizia-postale. [Online; accessed 10-October-2022]. 2022.

[14] Polizia postale. *Report 2020 della Polizia Postale: in aumento le minacce e le truffe sul web. +436% segnalazioni di fakenews*. https://www.interno.gov.it/it/notizie/report-2020-polizia-postale-aumento-minacce-e-truffe-sul-web-436-segnalazioni-fakenews. [Online; accessed 10-October-2022]. 2021.

[15] Shifa Rahaman. *A beginner's guide to image noise*. https://www.pixop.com/blog/image-noise-causes. [Online; accessed 19-October-2022]. 2021.

[16] Dasara Shullani et al. «VISION: a video and image dataset for source identification». In: *EURASIP Journal on Information Security* 2017.1 (ott. 2017), p. 15. ISSN: 2510-523X. DOI: 10.1186/s13635-017-0067-2. URL: https://doi.org/10.1186/s13635-017-0067-2.

[17] Mayank Tiwari e Bhupendra Gupta. «Image features dependant correlation-weighting function for efficient PRNU based source camera identification». In: *Forensic Science International* 285 (2018), pp. 111–120. ISSN: 0379-0738. DOI: https://doi.org/10.1016/j.forsciint.2018.02.005. URL: https://www.sciencedirect.com/science/article/pii/S0379073818300574.

[18] Wikipedia contributors. *Fast Fourier transform — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Fast_Fourier_transform&oldid=1114804913. [Online; accessed 1-November-2022]. 2022.

[19] Wikipedia contributors. *Gaussian noise — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Gaussian_noise&oldid=1074696378. [Online; accessed 19-October-2022]. 2022.

[20] Wikipedia contributors. *Salt-and-pepper noise — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Salt-and-pepper_noise&oldid=1085283109. [Online; accessed 19-October-2022]. 2022.

[21] Wikipedia contributors. *Shot noise — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Shot_noise&oldid=1114127133. [Online; accessed 19-October-2022]. 2022.