

# Image Forensics(Parte 3)

In questo capitolo, invece, parleremo di tecniche che permettono l'individuazione di manipolazioni non visualmente osservabili ma che alterano specifiche statistiche al livello di pixel o al livello semantico.

Vedremo i seguenti metodi:

- Cloning
- Resampling
- Social Media Upload

Infine, studieremo metodi basati sul Machine Learning.

## Cloning o Copy-move Forgery:

Qui, una parte dell'immagine viene copiata e incollata da un'altra parte della stessa immagine. Viene fatta per nascondere una scena della immagine o duplicarla.

Ma poichè si sta copiando una parte dell'immagine, le sue statistiche saranno uguali al resto dell'immagine.

I rilevatori di splicing sfruttano l'incoerenza delle caratteristiche statistiche locali, ma le caratteristiche locali delle regioni copiate sono coerenti con le altre parti.

Quindi, ci sarà nell'immagine contraffatta una correlazione tra la parte originale della foto e quella incollata.

**IDEA:** è quella di trovare regioni simili o caratteristiche che occorrono frequentemente nella foto.

Ci sono due approcci:

- Block-matching based
- Keypoints-based

## Block-Matching Based:

L'immagine viene scannerizzata dal l'angolo in alto a sinistra fino all'angolo in basso a destra (con blocchi di dimensione  $b \times b$ ). Dove il compito è trovare blocchi duplicati e connessi. Invece dell'intera regione duplicata.

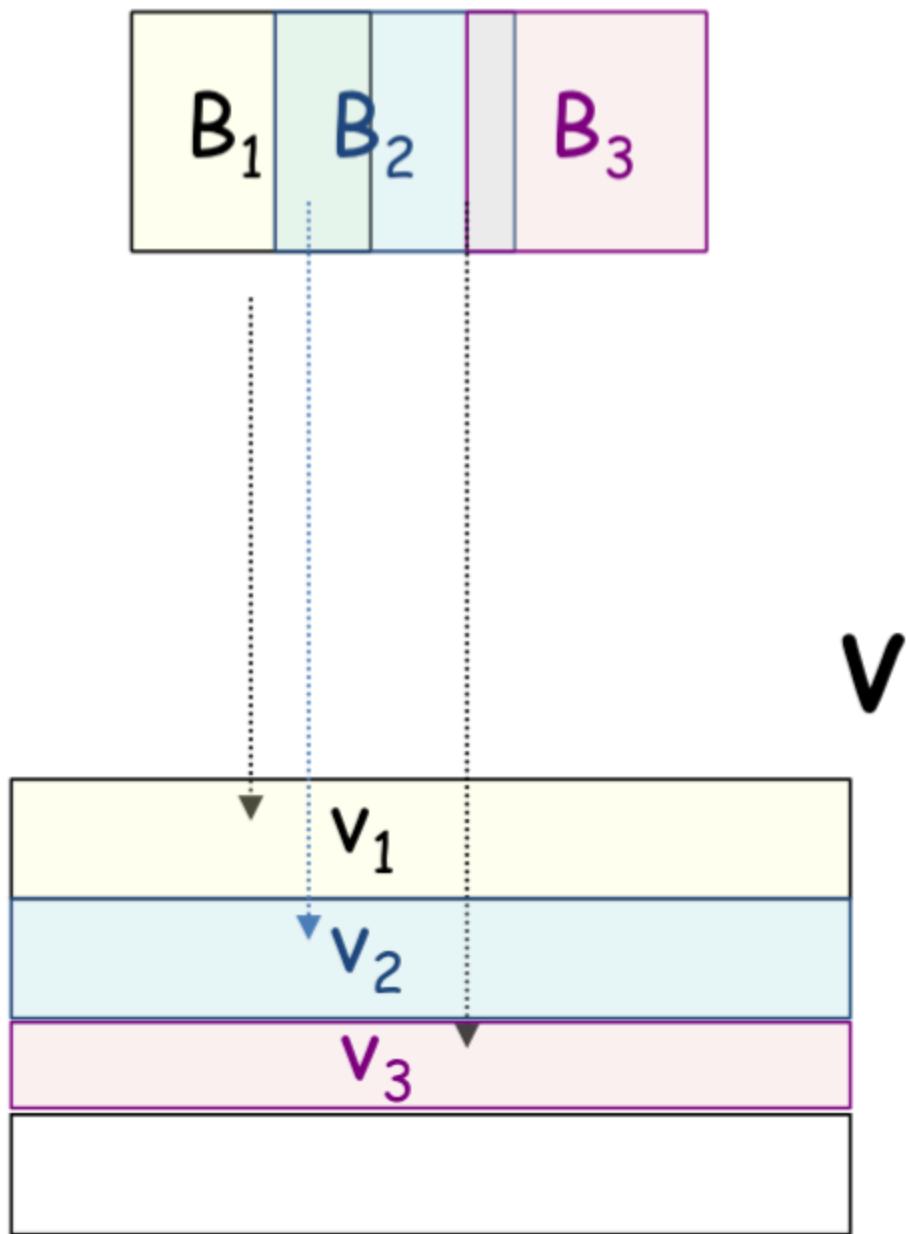
Come si fa, quindi a riconoscere, se una regione di una foto è stata copiata e incollata in un'altra parte della stessa foto?

Sostanzialmente prendiamo un blocco di dimensione  $b \times b$ , e facciamo una ricerca che stabilisca numerose coppie che hanno caratteristiche in comune. Queste ricerche però devono comprendere anche i blocchi che si sovrappongono. Ognuno di queste coppie di blocchi deve avere la stessa distanza poichè **la zona contraffatta è stata shiftata e quindi ogni blocco li dentro ha la stessa distanza da quella originale.**

Un'immagine  $M \times N$  è divisa in  $N_b = (M - b + 1) \times (N - b + 1)$ , ovvero il numero di blocchi  $b \times b$  sovrapposti nell'immagine. L'insieme di caratteristiche **di ogni singolo blocco** è vettORIZZATO e inserito in una matrice  $V$ .

$V$  è composto quindi:

- $N_b$  righe: che rappresentano ogni singolo blocco.
- F colonne: che rappresentano ogni caratteristica per confrontare il blocco.(intensità pixel...).



Queste righe sono riordinate *Lessicograficamente* ovvero si riordinano in base alle righe simili.

Ci possono essere, però, dei blocchi matchati erroneamente (potrebbero essere vicini e quindi essere simili sicuramente) di conseguenza bisogna matcharli anche in base alle posizioni di esse.

In generale, le righe simili devono adesso essere accoppiate e confrontate riga per riga. Ma per determinare quanto due righe consecutive siano simili, si utilizza il parametro chiamato  $N_n$  che rappresenta il numero di righe adiacenti da confrontare.

$N_n$ : è la distanza massima tra le righe che vengono considerate "adiacenti" nel processo di confronto

Ad esempio:

- $N_n = 1$ , quindi una riga  $r_i$  viene confrontata solo con le righe immediatamente precedenti e successive, cioè:

$r_{i-1}, r_i, r_{i+1}$

- La condizione  $|i - j| < N_n$  significa che una riga  $r_i$  sarà confrontata con tutte le righe  $r_j$  la cui distanza  $|i - j|$  è minore di  $N_n$ .

Per trovare i blocchi simili:

- ogni coppia di righe  $r_i$  e  $r_j$ ,  $|i - j| < N_n$ , salviamo la coordinata del blocco in una lista.
- Il vettore di shift  $s$  tra due blocchi accoppiati è calcolato come:

$$\underline{s} = (s_1, s_2) = (i_1 - j_1, i_2 - j_2)$$

- Per ogni distanza  $\underline{s}$ , si incrementa un contatore relativo a quello shift:  $C(\underline{s}) = C(\underline{s}) + 1$
- Questo contatore all'inizio è inizializzato a 0.

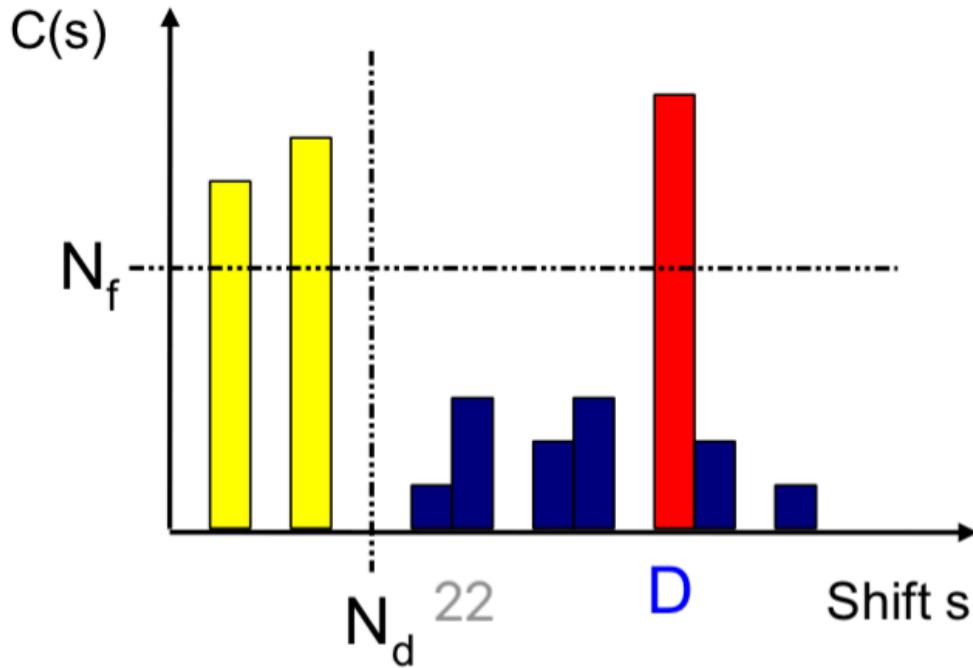
Quando tutte le coppie sono state valutate, il contatore  $C$  indica la frequenza di shift tra coppie simili di blocchi.

Di queste, però, bisogna:

- scartare i blocchi più vicini:  $s < N_d$
- scartare i blocchi che **sono sparsi ma con solito shift**:  $C(s) < N_f$
- i blocchi accoppiati che contribuiscono a quello specifico shift tale che  $C(s(i)) > N_f$  sono identificati come blocchi duplicati.

Il valore  $N_f$  è relativo alla dimensione dell'area più piccola che può essere identificato dall'algoritmo.

Mentre  $N_d$  è relativo alla minima distanza tra aree copiate.



**PROBLEMA:** bisogna trovare una rappresentazione robusta per blocchi di immagini, l'ampiezza dei pixel non sono caratteristiche robuste.

## Coefficienti DCT Quantizzati

Anche qui il numero di caratteristiche per ogni blocco è  $b \times b$  ovvero  $b^2$ . La quantizzazione permette di avere stessi valori DCT ma non uguali blocchi di immagini.

Ogni coefficiente DCtT è quantizzato con un elemento della tabella di quantizzazione Q di dimensioni  $b \times b$ .

Il fattore Q controlla la sensitività dell'algoritmo nel grado di corrispondenza tra blocchi. (più è quantizzato più match erronee potranno esserci).

Il blocco di dim B e il limite  $N_f$  controllano la minima dimensione del clone da individuare.

## PCA (Principal Components Analysis)

Il PCA serve a ridurre la complessità di dati a dimensione alta in un sottospazio con dimensione più basso.

Così facendo la dimensione della matrice V è ridotta minimizzando la perdita di informazione, aumentando la velocità di computazione.

Sia  $\mathbf{x}_i$ ,  $i = 1, \dots, N_b$  rappresentano le colonne della matrice V e che quindi in ogni colonna di una riga sono i pixel di ogni blocco immagine.

La media è:

$$\mu = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$$

Si calcola dati a media nulla impacchettati in una matrice di dim  $B \times N_b$ :

$$M = (\mathbf{x}_1 - \mu, \mathbf{x}_2 - \mu, \dots, \mathbf{x}_N - \mu)$$

La matrice di covarianza viene poi calcolata in questo modo:

$$C = MM^T$$

I principali componenti sono gli autovettori  $e_j$  della matrice C che corrispondono agli autovalori  $\lambda_j$ :  $C\vec{e}_j = \lambda_j e_j$  con  $j=1, \dots, B$

Gli autovettori servono per ricreare ogni valore  $\vec{x}_i = \sum_{j=1}^B a_j \vec{e}_j$

dove:  $a_j = \mathbf{x}_i^T \mathbf{e}_j$  con  $a_j$  che è la nuova rappresentazione di ogni blocco.

La dimensionalità di ogni  $\mathbf{x}_i$  è ridotta da B a p proiettando ogni  $x_i$  ai p autovalori-autovettori più **GRANDI**.

Di conseguenza il vettore p-dimensionale  $\mathbf{a}_i = (a_1, \dots, a_p)$  facendo diventare il numero delle caratteristiche  $F = p << b^2$ .

## SIFT-Based:

La complessità della ricerca di regioni clonate si può ridurre facendo operazioni sulle caratteristiche salienti dell'immagine.

Vengono utilizzate caratteristiche visive locali come SIFT(Scale Invariant Feature Transform) per la loro robustezza.

Le parti copiate hanno la stessa apparenza di quelle originali, quindi i keypoint estratti dalla copia saranno simili a quelli originali.

Le caratteristiche SIFT servono:

- trovare la parte copiata.
- trovare la trasformata geometrica che è stata applicata.

## Detection della parte copiata:

Queste caratteristiche visive locali hanno bisogno di due passi:

- Individuazione di questi punti (keypoints), sono punti di interesse come angoli, spigoli o regioni ad alto contrasto,invarianti a trasformazioni in scala, rotazione e illuminazione. Un pixel è considerato un punto di interesse se è maggiore o minore di tutti i suoi vicini nelle 8 direzioni nel piano e nei 9 livelli sopra e sotto nella piramide di scale (totale 26 vicini).

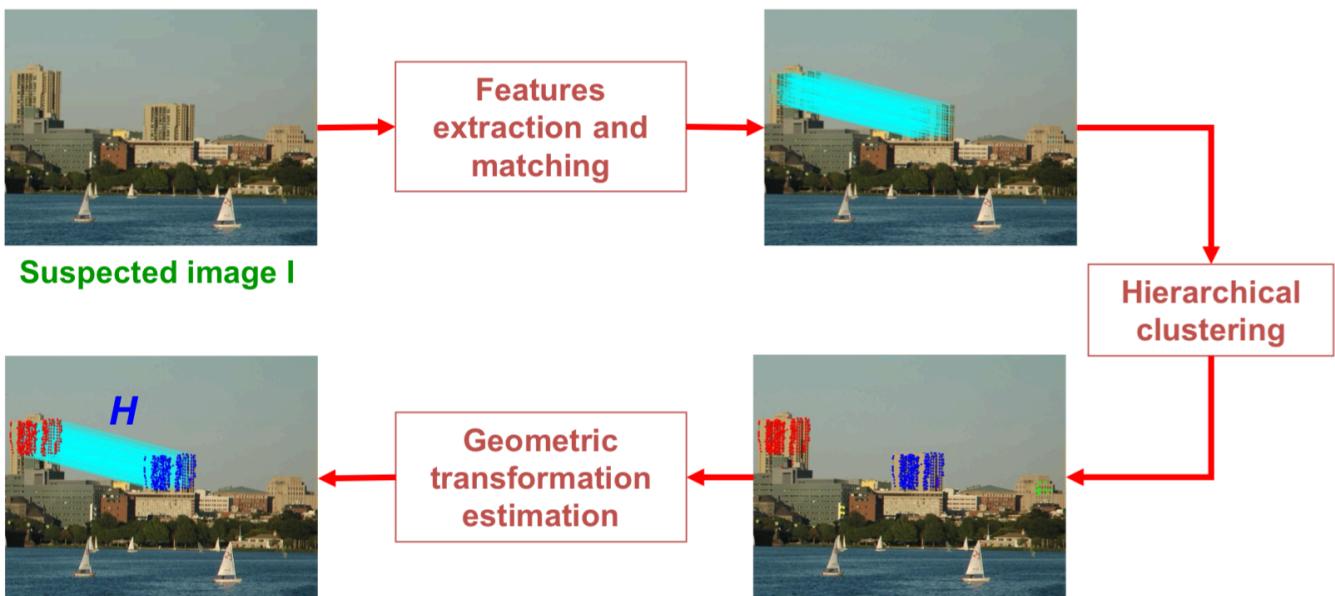
- Per ogni punto, viene costruito un robusto descrittore locale.

Quindi, per ogni keypoint viene costruito un vettore di caratteristiche.

$$\mathbf{x}_i = (x, y, \sigma, o; \mathbf{f})$$

dove:

- $x, y$  sono coordinate spaziali.
- $\sigma$  è la scala di un keypoint
- $o$  è la rotazione di un keypoint
- $\mathbf{f}$  è il SIFT descriptor (128 elementi)



I keypoints  $X = (x_1, \dots, x_N)$  sono estratti con i loro descrittori SIFT. Il miglior candidato per ogni keypoint  $x_i$  è trovato identificando il suo vicino tra gli  $n-1$  keypoints che **ha la minima distanza Euclidea tra i descrittori**.

Per migliorare la robustezza, si può adottare il rateo tra il primo più vicino e il secondo.

Due keypoints sono corrisposti se il rateo  $d_1/d_2 < T$ .

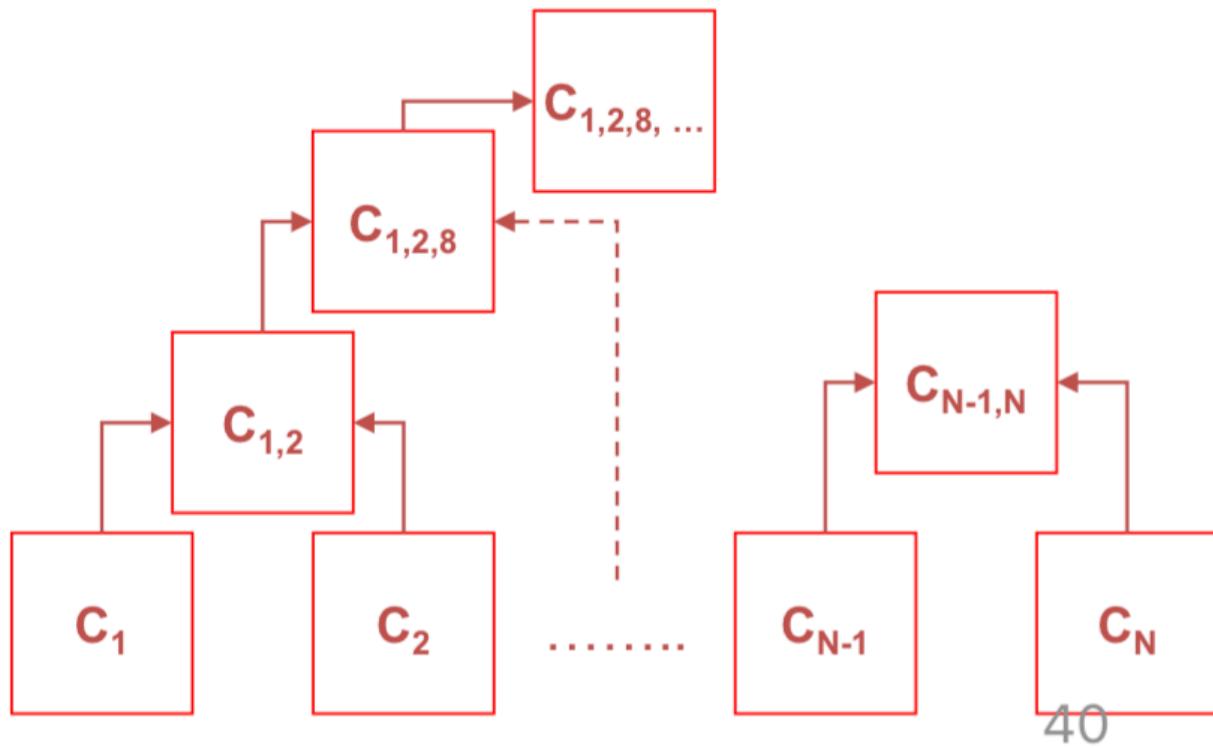
Si itera per ogni keypoint in  $X$  e si ottiene un insieme di punti matchati. Tutti i punti matchati sono tenuti, mentre quelli isolati sono scartati.

Per trovare le possibili aree clonate usiamo il **Clustering gerarchico agglomerativo**, ovvero raggruppare in base alla locazione spazioale dei punti matchati.

Il raggruppamento viene fatto attraverso una struttura ad albero.

Si inizia attribuendo un cluster ad ogni keypoint e poi si computa tutte le distanze spaziali reciproche tra questi cluster.

Due cluster con la minima distanza sono uniti.



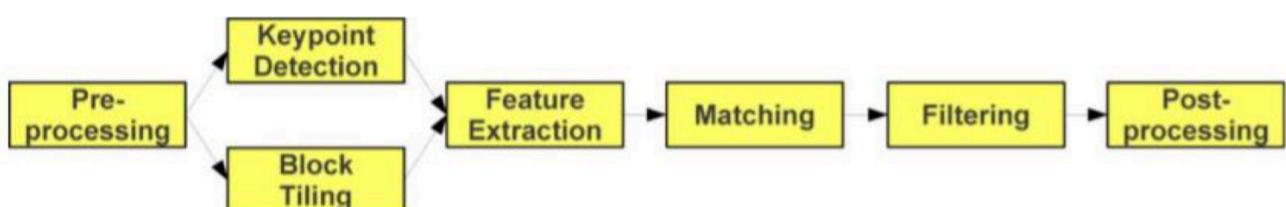
Dopo di che vengono eliminati i cluster che non contengono un numero significativo di keypoint matchiati.

## Stima della trasformazione geometrica:

La stima viene fatta con l'algoritmo RANSAC (RANdom SAmple Consensus) il quale permette di migliorare i risultati riducendo gli effetti di disturbo dati dai valori anomali.

Le trasformazioni di traslazione, rotazione e scaling tra l'originale e quella copiata possono essere determinati usando un insieme di matched point estratti.

## Considerazione :



I limiti dei due approcci sono rispettivamente:

- **Block Matching Based** → lento, le caratteristiche non sono robuste per la trasformazione geometrica. Robustezza vuol dire che riesce a rilevare correttamente nonostante disturbi della scena.
  - **Keypoints-Based** → veloce, ma non utile nelle scene copiate **omogenee e lisce**.
- 

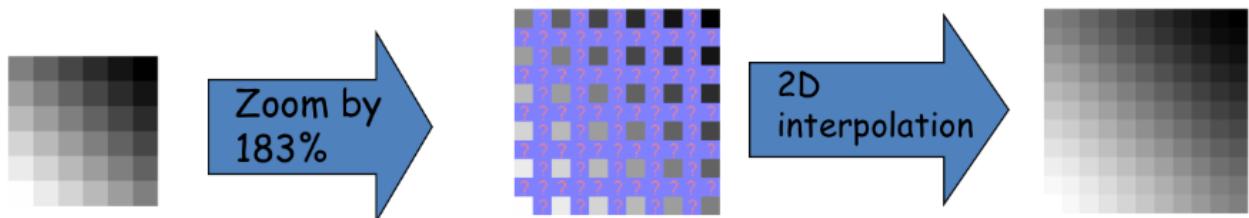
## Resampling

Con resampling si intende ricampionamento, per fare un fotomontaggio convincente bisogna ridimensionare, ruotare e stretchare la porzione d'immagine da incollare.

Ricordiamo che l'immagine digitale è formato da pixel fissati in una certa griglia fissata, se però si fanno delle modifiche geometriche **modificando** le posizioni dei pixel originali → si ha la creazione di nuovi valori per i pixel.

Ad esempio: Se ridimensiono una foto , tipo facendo uno zoom, allora aumento dei pixel in quella zona della foto che prima non c'erano e quindi alcuni pixel vengono creati.

Questa creazione si fa tramite l'**interpolazione** e serve per approssimare questi pixel nuovi basando il valore sui pixel vicini.



In generale, \*\*gli algoritmi di interpolazione creano delle dipendenze periodiche tra gruppi di campionamenti vicini.

## Elementi Introduttivi Resampling:

Consideriamo un segnale campionato  $f(x)$  1-D (quindi non una foto) con  $m$  campionamenti.  $f(x)$  è convertito in un nuovo segnale  $g(x)$  con  $n$  campionamenti, attraverso un fattore di scallo  $p/q$ .

Quindi:  $n = \lfloor m \cdot p/q \rfloor$

### 1. Up-sample

- Si crea un nuovo segnale  $f_u(x)$  con  $p \cdot m$  campioni, dove  $p = 4$ .

- In  $f_u(x)$ , si mantengono i valori originali nei punti multipli di p (ad esempio 4, 8, 12, ecc.) e si pongono a 0 tutti gli altri punti.

## 2. Interpolazione

- Si applica un filtro passa-basso  $h(x)$  a  $f_u(x)$ . Questo filtro riempie i campioni “vuoti” (valori a zero) con nuovi valori interpolati in modo da rendere il segnale continuo.
- La convoluzione è rappresentata da:  $f_i(x) = f_u(x) \otimes h(x)$
- $h(x)$ : un filtro, come interpolazione lineare o cubica.
- Questo step aggiunge i valori non nulli alle posizioni intermedie.

## 3. Down-sample

- Dopo l'interpolazione, il segnale  $f_i(x)$  viene ridotto al numero desiderato di campioni (n).
- Si preleva un campione ogni q (ad esempio 3) dalla versione interpolata:  $g(x) = f_i(qx)$ , per  $x = 1, 2, \dots, n$ .

Considerazioni:

- $f(px)$  : rappresenta l'up-sampling, dove si aumentano i campioni mantenendo l'indice originale e aggiungendo 0 nei punti vuoti.
- $f(qx)$  : rappresenta il down-sampling, dove si prelevano solo alcuni campioni (ogni q) dal segnale originale.

Questi tre passi descritti, sono lineari quindi si può scrivere come un equazione lineare:

$$\vec{g} = \mathbf{A}_{p/q} \cdot \vec{f}$$

dove  $f$  è di dimensione m,  $g$  di dimensione n quindi A sarà di dimensione  $n \times m$ .

Si nota che il Resampling introduce correlazioni tra i campionamenti vicini questa correlazione dipende sostanzialmente dal **resampling rate** e il filtro  $h(x)$ .

Quindi il resampling lascia una traccia. Quando questa viene trovata allora questa è una prova di una manipolazione.

In particolare abbiamo:

$$g_i = \sum_{k=-N}^N \alpha_k \cdot g_{i+k}$$

si può riscrivere in questo modo:

$$g_i - \sum_{k=-N}^N \alpha_k \cdot g_{i+k} = 0$$

dove  $g_i$  rappresenta una riga per colonna di  $A_{p/q} \cdot \vec{f}$  (anche  $g_{i+k}$ ) quindi è inteso come:

$$\mathbf{a}_i \cdot \mathbf{f} - \sum_{k=-N}^N \alpha_k \cdot (\mathbf{a}_{i+k} \mathbf{f}) = 0$$

raggruppando  $\mathbf{f}$  viene:

$$f \cdot (\mathbf{a}_i - \sum_{k=-N}^N \alpha_k \cdot (\mathbf{a}_{i+k})) = 0$$

quindi:

$$\mathbf{a}_i - \sum_{k=-N}^N \alpha_k \cdot (\mathbf{a}_{i+k}) = 0$$

Se la forma della correlazione è conosciuta ,cioè si conoscono i valori  $\alpha$  e  $N$  , allora è possibile determinare quale pixel è stato ricampionato. (ovvero quello che soddisfa questa equazione).

Se sono conosciuti i pixel correlati con i suoi vicini allora è possibile invece trovare la dimensione del "vicinato" e l'insieme dei coefficienti che soddisfano questa correlazione.

---

## Resampling Detection

Data un immagine da analizzare non sappiamo:

- ne i pixel correlati (quindi quelli creati).
- ne la forma della correlazione.

Una possibile soluzione è quella di sviluppare un algoritmo *expectation/maximization* per stimare simultaneamente:

- insieme periodico di campionamenti correlati ai suoi vicini.
- una forma specifica di questa correlazione ( $\alpha$ ).

Ipotizziamo che ogni pixel  $g_i$  appartenga ad uno dei due modelli:

- $M_1$ :  $g_i$  è linearmente correlato ai suoi vicini, ovvero è generato da I seguente modello.

$$g_i = \sum_{k=-N}^N \alpha_k g_{i+k} + n(i)$$

dove  $n(i)$  è una distribuzione Gaussiana con media nulla e varianza  $\sigma^2$ , mentre  $\alpha$  sono i parametri specifici della correlazione.

- $M_2$ :  $g_i$  non è correlato con i suoi vicini. Cioè è uniformemente distribuito su tutti i possibili

valori.

Vediamo quindi la probabilità che l'evento  $g_i$  accada assumendo per vero che  $g_i \in M_1$  :

$$P(g_i|g_i \in M_1) = P(n(i)) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(g_i - \sum_{k=-N}^N \alpha_k g_{i+k})^2}{2\sigma^2}}$$

se si scrive all'esponente:  $r(i) = (g_i - \sum_{k=-N}^N \alpha_k g_{i+k})^2$  così diventa:

$$P(g_i|g_i \in M_1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{r_i^2}{2\sigma^2}}$$

dove quindi  $r_i$  è la differenza tra il valore  $g_i$  e la stima compiuta attraverso i vicini.

Altrimenti con il modello  $M_2$  assumiamo sia uniformemente distribuita quindi:

$$P(g_i|g_i \in M_2) = 1/\Delta$$

dove  $\Delta$  rappresenta l'intervallo dei valori interni che i campionamenti possono assumere.

---

Utilizziamo quindi EM Algorithm, che è un algoritmo iterativo a due passi:

- *Expectation (E) step*: è stimata la probabilità di ogni campionamento che appartiene al modello stimato.
- *Maximization (M) step*: sono stimati, invece, la forma della correlazione tra i campionamento ( $\alpha$ ) e la varianza.

Lo step E richiede comunque una stima di  $\alpha \rightarrow$  prima iterazione è scelta randomicamente.

Specificatamente nel passo E si calcola la probabilità che ogni campionamento appartenga a  $M_1$  ovvero:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j)P(A_j)},$$



$$P(B)$$

$$\Pr\{g_i \in M_1 | g_i\} = \frac{\Pr\{g_i|g_i \in M_1\} \Pr\{g_i \in M_1\}}{\Pr\{g_i|g_i \in M_1\} \Pr\{g_i \in M_1\} + \Pr\{g_i|g_i \in M_2\} \Pr\{g_i \in M_2\}}$$

Cerchiamo di semplificare questa equazione, sapendo che  $P(g_i \in M_1) = P(g_i \in M_2) = 1/2$

$$Pr\{g_i \in M_1 | g_i\} = \frac{Pr\{g_i | g_i \in M_1\}}{Pr\{g_i | g_i \in M_1\} + Pr\{g_i | g_i \in M_2\}}$$

$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-r_i^2/2\sigma^2}$  ←  $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-r_i^2/2\sigma^2}$   
 ↓  $\frac{1}{\Delta}$   
 $\Pr\{g_i \in M_1 | g_i\} = \frac{e^{-r_i^2/2\sigma^2}}{e^{-r_i^2/2\sigma^2} + \sqrt{2\pi\sigma^2}/\Delta}$

Quest'ultima formula è calcolata stimando il residuo, quindi possiamo scrivere:

$$P(g_i \in M_1 | g_i) = P(\alpha | r_i) = w(i)$$

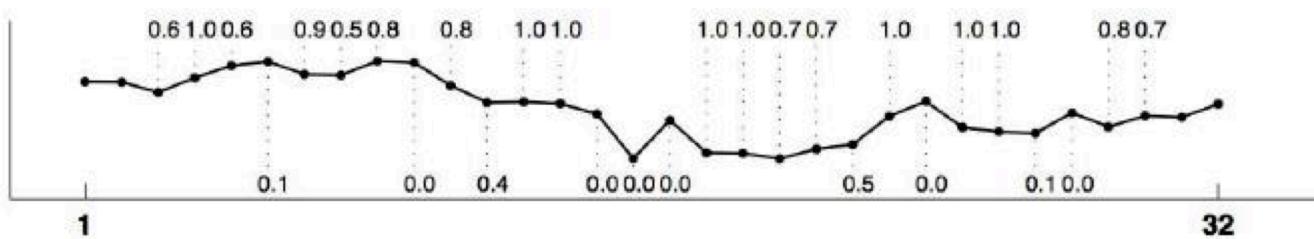
Mentre per quanto riguarda il passo M la nuova stima di  $\alpha$  è compiuta minimizzando la funzione errore quadratico medio:

$$E(\alpha) = w(i)(g_i - \sum_{k=-N}^N \alpha_k g_{i+k})^2$$

dove ricordiamo che  $w(i) = P(g_i \in M_1 | g_i)$  e  $\alpha_0 = 0$ .

La funzione di errore quadratico si minimizza facendo il gradiente rispetto a  $\alpha$  e impostando il risultato uguale a 0. Questi due passi vengono eseguiti finché non si ha un  $\alpha$  stimato stabile.

Guardiamo un esempio:

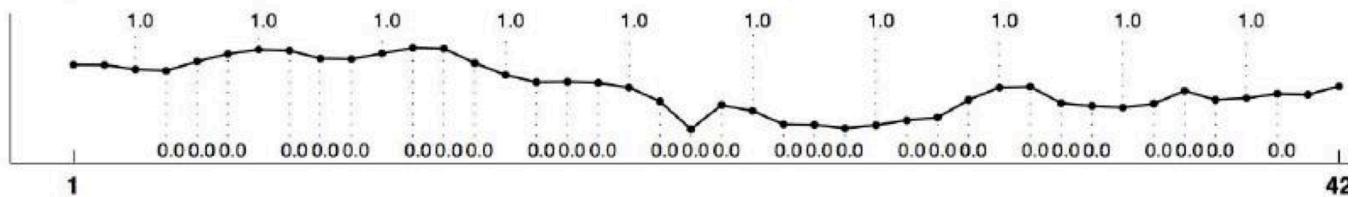


Questo è un risultato dell'algoritmo EM su un segnale originale (non ridimensionato)  $f$ .

Ogni campionamento, nella figura, è segnato con la sua probabilità di essere correlato.

Vediamo quindi che la correlazione non segue una certa forma.

Invece, adesso guardiamo quest'altro risultato dove la funzione  $f$  è stata ridimensionata.



Si nota che ogni 4 campionamento se ne ha uno che siamo sicuri sia correlato ai suoi vicini.

---

Questo pattern periodico è dato proprio dal ricampionamento, poichè esso è sempre causa di effetti di periodicità.

Attenzione però perchè ci sono dei parametri che introducono dei pattern periodici indistinguibili l'uni dagli altri. Ma questo non vuol dire che non è rilevabile il campionamento solo che è difficile la stima del fattore.

Ci sono però un particolare tipo di ricampionamento che **non introduce correlazioni periodiche rilevabili**. Ad esempio un **down-sampling senza interpolazione**, come nella figura sotto, che fa sì che le righe non possono essere riconducibili a combinazioni lineari.

$$A_{1/2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ & & & & \ddots \\ & & & & \ddots \end{bmatrix}$$

**Resampling 2-D:**

Si può estendere il ragionamento analogo sulla situazione immagini. Avrà i soli 3 step.

$x_1$	$x_2$
$x_3$	$x_4$

$x_1$	0	$x_2$
0	0	0
$x_3$	0	$x_4$

$g_1$	$g_2$	$g_3$
$g_4$	$g_5$	
$g_7$		$g_9$

Dove si assume l'interpolazione lineare in questo modo:

$$g_2 = 0.5g_1 + 0.5g_3$$

$$g_4 = 0.5g_1 + 0.5g_3$$

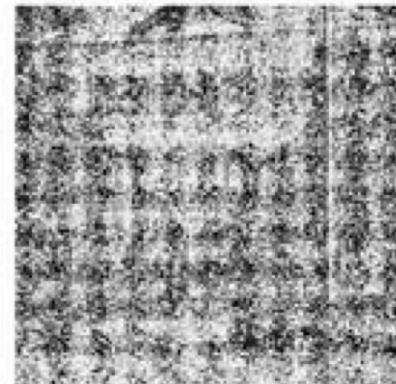
$$g_5 = 0.25g_1 + 0.25g_3 + 0.25g_7 + 0.25g_9$$

Dove, quindi, i pixel che stanno nelle colonne righe dispari e colonne pari saranno sottoposte ad una combinazione lineari con i vicini orizzontali.

Altrimenti, i pixel che stanno nelle colonne pari e righe dispari saranno correlati con i vicini verticali.

Si può utilizzare EM anche per 2-D solo che il risultato sarà **una p-map** dove in questo risultato i pixel bianchi denotano una alta probabilità di essere correlati, mentre quelli neri denotano una bassa probabilità.

Image

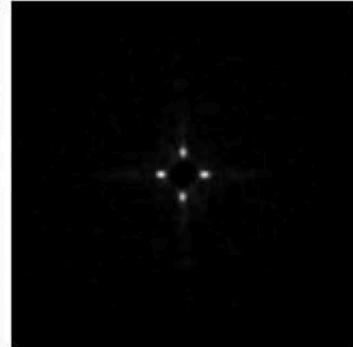
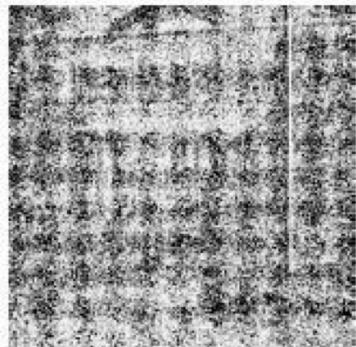


p-map

La periodicità può essere valutata meglio attraverso la trasformata di Fourier, la presenza di picchi, infatti, rivela la presenza di correlazioni periodiche nell'immagine. Poiché lo spettro

individua proprio dei comportamenti periodici.

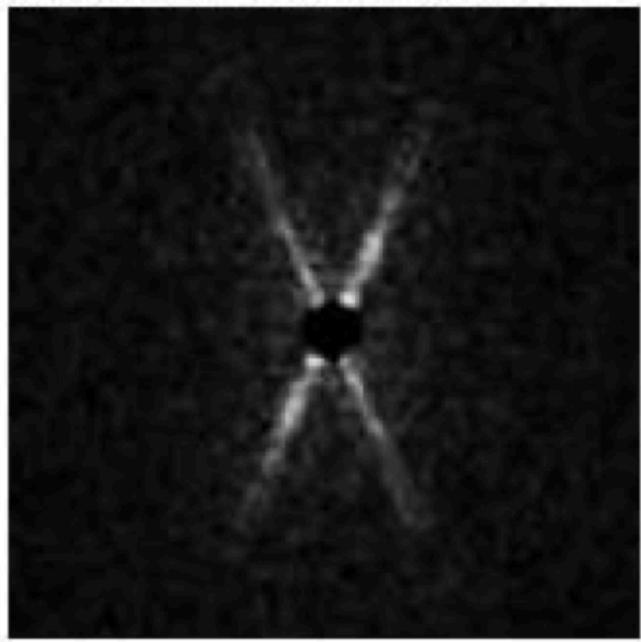
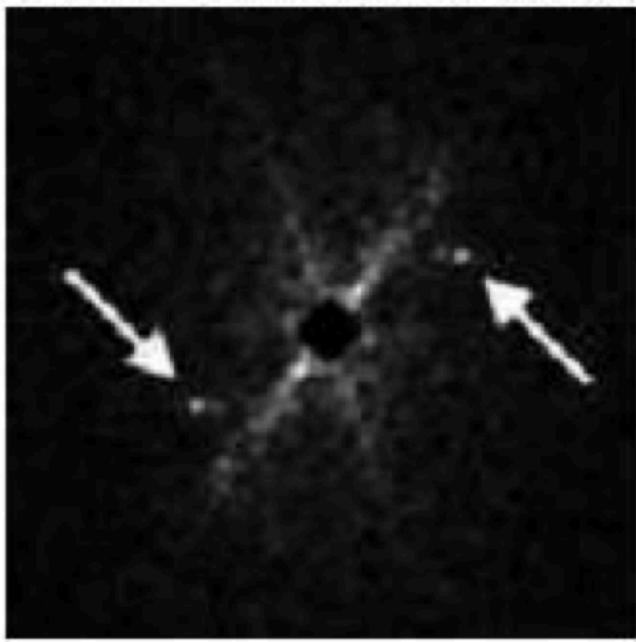
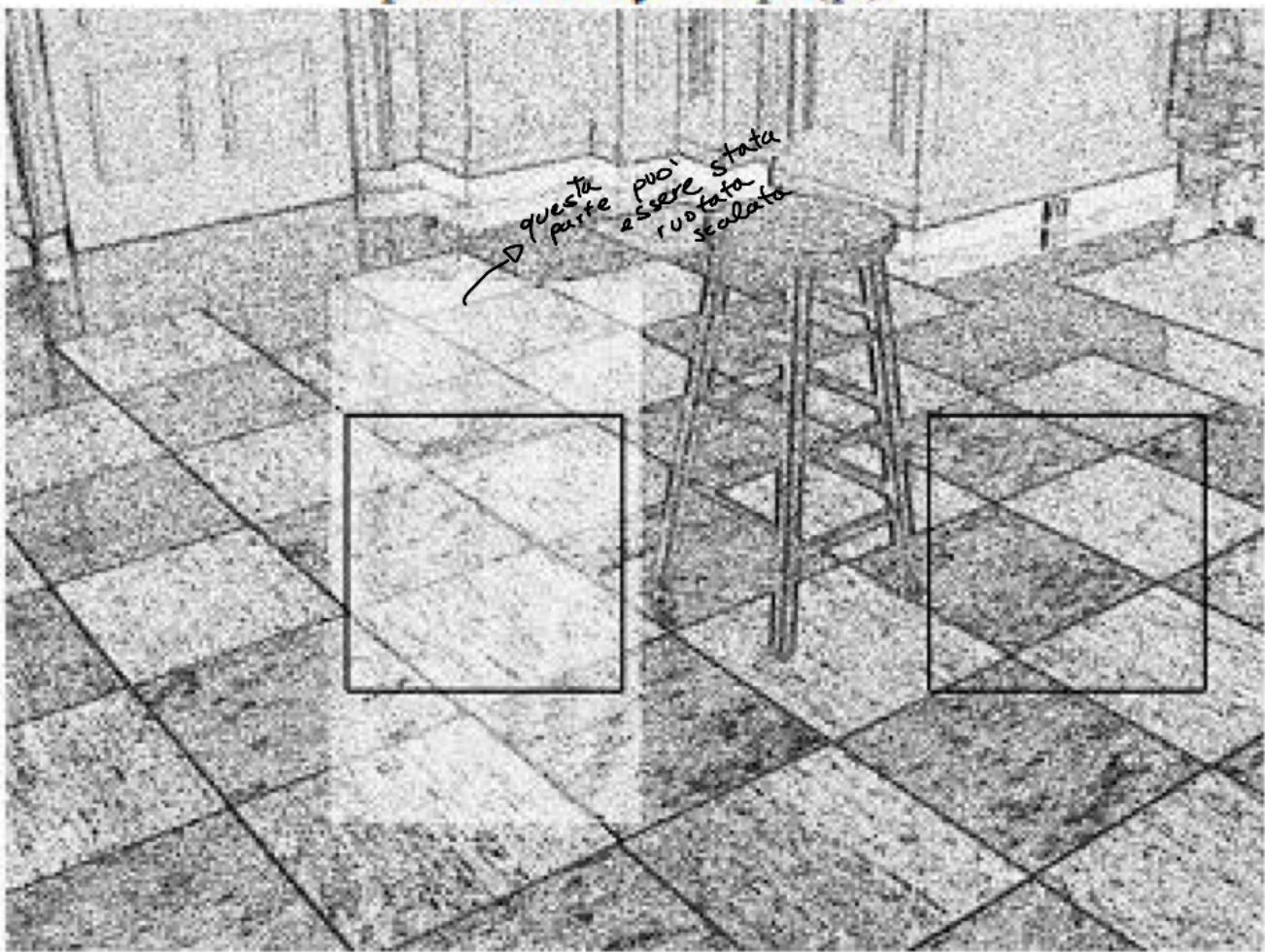
P-map



Fourier  
transform  
of p-map

Questo metodo serve anche per quelle manomissioni dove si ripete una parte dell'immagine per nasconderne un'altra. Si può individuare proprio dalla trasformata di Fourier.

## probability map ( $p$ )



Poichè sappiamo che quella parte dell'immagine è stata sicuramente ruotata e ridimensionata avendo quindi subito un ricampionamento.

# Identificazione della sorgente da immagini condivise via Social:

Molti utenti generano foto e video che vengono caricate e condivisi sui social network e essere in grado di capire se un'immagine è stata scaricata da una foto o direttamente da una fotocamera digitale può essere cruciale per delle investigazioni consecutive.

(Se è stata scaricata da un altro social → vuol dire che alla peggio è stata ritoccata e ripubblicata ).

Un approccio è quello di usare le reti neurali convuzionali (CNN) per determinare la provenienza dell'immagine.(social network, applicazione Messaggistica o direttamente dalla fotocamera)

Senza ricorrere a: EXIF, dimensione del file e nome, risoluzione dell'immagine.

Identificando alcune tracce distintive "inevitabilmente" impresse in ciascun contenuto digitale. durante il processo di caricamento / download da parte di ogni specifico social network.  
Usando Deep Learning per eseguire la classificazione.

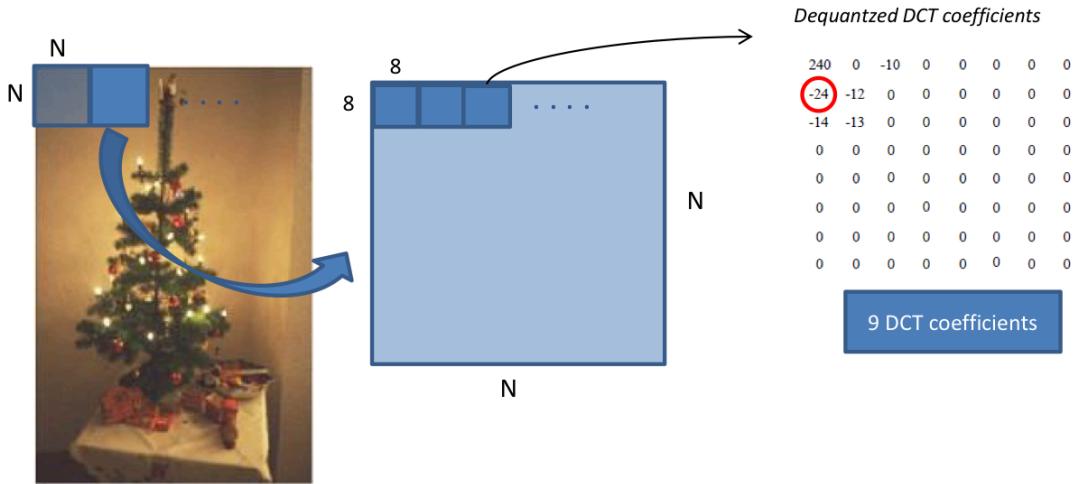
(Da notare che qui non si vuole fare un'indagine incrociata con il NoisePrint, ma si vuole solo riconoscere da dove è stata scaricata)

Ecco un riassunto chiaro del contenuto delle slide, basato sulle immagini che hai condiviso:

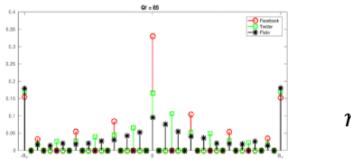
## Metodo Proposto

Il metodo proposto si concentra sull'analisi a livello di patch di immagini ed è costituito da due moduli principali:

- **Estrazione di caratteristiche basate su DCT:**
- Le immagini sono suddivise in patch di dimensioni senza sovrapposizioni.
- Vengono calcolati i coefficienti DCT per ogni blocco all'interno di ogni patch. Tra questi, vengono selezionati i primi 9 coefficienti di frequenza spaziale (oltre al coefficiente DC), seguendo l'ordine zig-zag.
- L'analisi DCT produce un insieme di 909 caratteristiche (101 istogrammi per ciascun coefficiente DCT).



For each DCT coefficient at position  $(i, j)$  in a block  $8 \times 8$ , the occurrences of the value  $m$  are counted and their histogram is built.



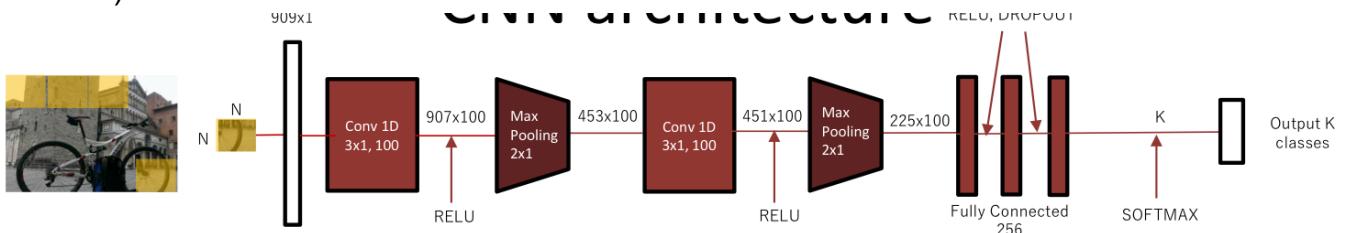
$$h_{(i,j)}(m)$$

$$m = \{-50, 0, +50\}$$

909 features =  
101 histogram bins x 9 coeff. DCT

#### • Architettura CNN:

- Una rete convoluzionale 1D prende in input i vettori di caratteristiche (909 elementi).
- La rete include convoluzioni seguite da max-pooling per ridurre la dimensionalità e tre livelli completamente connessi (due di dimensione 256 e uno con classi come output).
- L'attivazione è gestita tramite la funzione ReLU, mentre il livello di output utilizza Softmax per fornire la probabilità di appartenenza a ciascuna delle classi (ad esempio, diversi social network).



La convoluzione in questo metodo funziona su un input costituito da un vettore di dimensione fissa (909 elementi), che rappresenta i coefficienti DCT e gli istogrammi delle patch. L'obiettivo è identificare modelli locali significativi che aiutano a distinguere le immagini in base alla loro origine (social network).

#### 1. Kernel (o filtro):

- La convoluzione utilizza piccoli vettori chiamati **kernel** (o filtri), che sono array di pesi.
- Ogni kernel scorre lungo il vettore di input (in modo simile a come un filtro scorre su un'immagine) e calcola un prodotto scalare tra il kernel e i valori locali del vettore.

#### 2. Output della convoluzione:

- Per ogni posizione del kernel sul vettore di input, si calcola un valore risultante.
- Questo produce un nuovo vettore (o "feature map") che evidenzia pattern rilevanti, come

variazioni o correlazioni tra i coefficienti DCT.

### 3. Struttura specifica nella rete:

- Nel caso della rete descritta nelle slide:
- I vettori di input di dimensione 909 passano attraverso **due strati di convoluzione 1D**
- Ogni convoluzione riduce dimensionalità e complessità, mantenendo solo le caratteristiche più importanti.
- **Max pooling** viene applicato dopo ogni convoluzione per ridurre ulteriormente la dimensionalità, selezionando i valori più rilevanti (massimi) in regioni locali del vettore.

### 4. Funzione di attivazione (ReLU):

- Dopo ogni convoluzione, viene applicata una funzione di attivazione (Rectified Linear Unit, o ReLU), che imposta a zero i valori negativi, introducendo non linearità e migliorando la capacità della rete di apprendere modelli complessi.

### 5. Finalità:

- La convoluzione permette alla rete di imparare rappresentazioni che catturano i dettagli più significativi dei coefficienti DCT estratti dalle patch. In questo modo, la rete riesce a classificare le immagini in base alla loro origine social, nonostante le variazioni di qualità o risoluzione.  
Cerchiamo di capire come fa ad imparare la rete...

La fa attraverso un momento di apprendimento nel quale si inizializzano i **kernel**, ovvero i pesi, con valori random. Esegue tutta la catena vista (Max Polling, funzione di attivazione) e si creano delle previsioni. La rete, poi, confronta l'etichetta corretta con la risultante usando una **funzione di perdita** (Cross-entropy loss che misura quanto l'etichetta generata sia lontana dalla verità).

La perdita calcolata verrà utilizzata per aggiornare i pesi dei filtri e delle connessioni e lo fa calcolando i gradienti della funzione di perdita rispetto a ciascun peso (*connessioni*). E poi vengono propagati partendo dall'ultimo strato fino ai primi. Gli strati sono il numero di stadi a cui è sottoposto il dato.

Una volta calcolati i gradienti si utilizza ottimizzatore per diminuire la perdita ed aumentare l'apprendimento (*Adam*). Ovviamente si fanno su nuovi dati (e tanti).

### Dataset Utilizzati\*

Sono stati utilizzati tre dataset principali:

- **UCID Social**: Include 1000 immagini UCID, trasformate in JPEG con diversi fattori di qualità (a), caricate/scaricate su Flickr, Facebook e Twitter.
- **Public Social**: Include 3000 immagini non controllate provenienti da Flickr, Facebook e Twitter, con variazioni di dimensione, contenuto e fattore di qualità JPEG.
- **IPLAB**: 1920 immagini appartenenti a 8 classi (240 per ciascuna classe), con risoluzioni diverse e distribuite su 5 social network e 2 app di messaggistica (WhatsApp e Telegram).

I dataset sono suddivisi in:

- **Training (80%)**
- **Validazione (10%)**
- **Testing (10%)**

### 3. Classificazione a Livello di Patch

I risultati della classificazione a livello di patch sono riportati per i dataset UCID e Public:

- **UCID:**

- Accuratezza media: **98,41%**.

- Classificazione altamente precisa per Flickr (99,79%) e Twitter (99,30%).

- **Public:**

- Accuratezza media: **87,60%**.

- Migliori risultati per Flickr (88,80%) e Twitter (89,78%), ma con una performance leggermente inferiore rispetto a UCID.

---

## Supervisionatore generico CNN:

Non stanno a guardare delle manipolazioni specifiche o artefatti ma queste CNN trovano i pixel creati identificando features anomale locali.

Un esempio ne è **Mantra Net**, il quale è capace di distinguere 385 tipi di manipolazioni ed è robusto anche alle manipolazioni più sconosciute.

---

Ma il problema con questa CNN generica è che è molto difficile addestrare una rete per qualsiasi tipo di manipolazione, per evitare questo problema si usa un approccio "one-class" ovvero addestro un modello con tutti dati "**pristini**"(non manipolati) così qualsiasi anomalia viene rilevata.

## Modello per vedere NoisePrint

Learning procedure: minimize [maximize] the distance between residual patches from same [different] cameras, position *AND editing history*

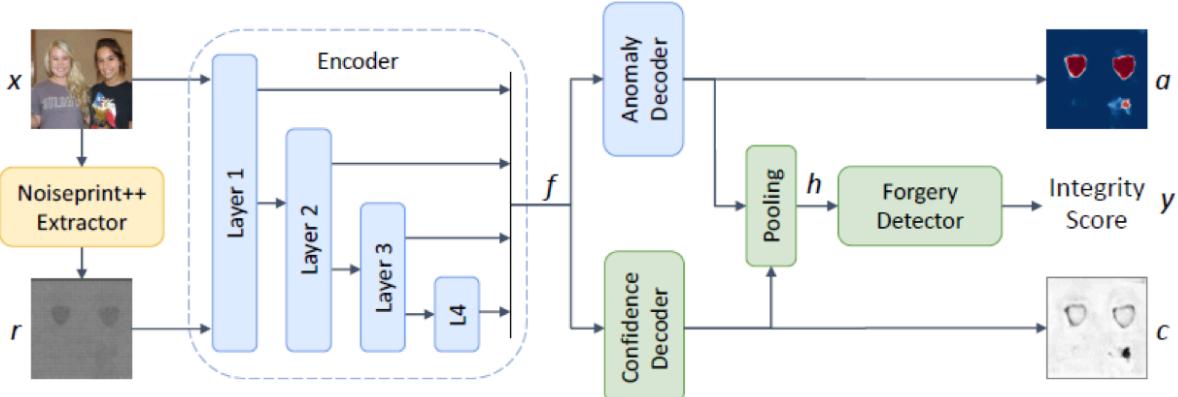
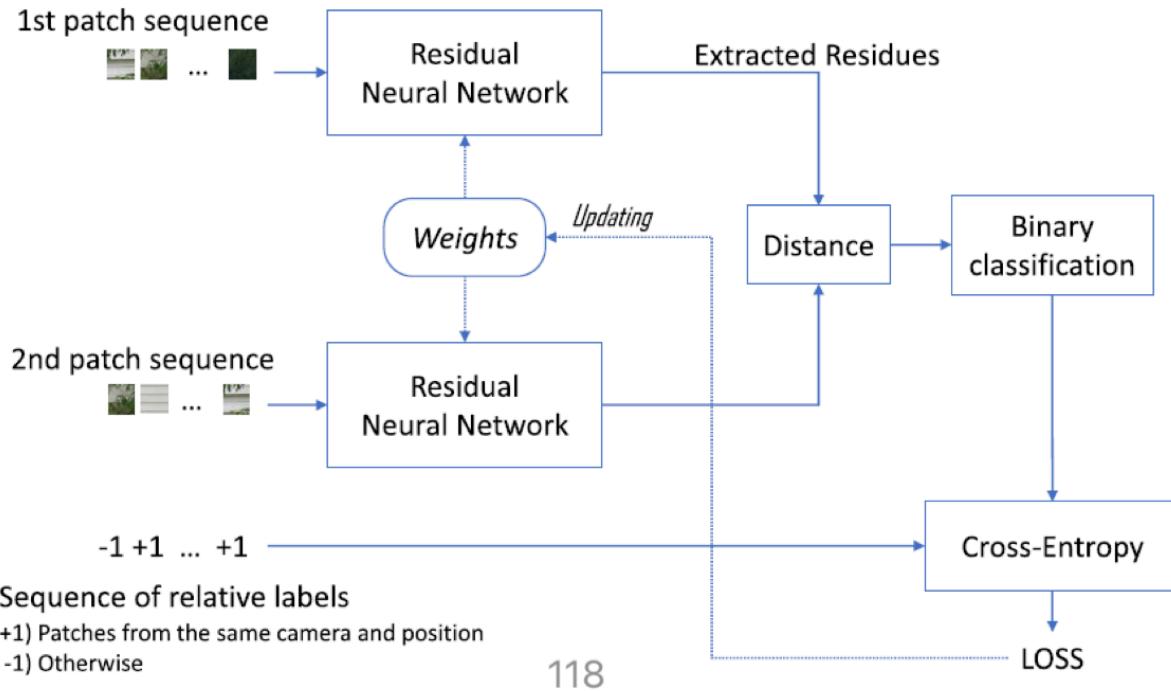


Figure 2. TruFor framework. The Noiseprint++ extractor takes the RGB image to obtain a learned noise-sensitive fingerprint. The encoder uses both the RGB input and Noiseprint++ for jointly computing the features that will be used by the anomaly decoder and the confidence decoder for pixel-level forgery localization and confidence estimation, respectively. The forgery detector exploits the localization map and the confidence map to make the image-level decision. The different colors identify the modules learned in each of the three training phases.